Universidade de Aveiro Departamento de
2011 Eletrónica, Telecomunicações e Informática

Elodie Ferreira

Plataforma de Gestão de Consultas Online para os SASUA

Management Platform for Medical Appointments for SASUA

Elodie Ferreira

Plataforma de Gestão de Consultas Online para os SASUA

Management Platform for Medical Appointments for SASUA

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Mestrado Integrado em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Osvaldo Manuel da Rocha Pacheco, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente

Prof. Dr. José Luís Guimarães Oliveira
professor associado da Universidade de Aveiro

Prof. Dr. Fernando Joaquim Lopes Moreira
professor associado da Universidade Portucalense

Prof. Dr. Osvaldo Manuel Da Rocha Pacheco
professor associado da Universidade de Aveiro

agradecimentos

Agradeço em primeiro lugar aos meus pais por me terem dado a possibilidade de tirar um curso superior, e por terem suportado a minha ausência em datas importantes por diversas vezes por ter sempre os estudos em primeiro lugar. Agradeço também ao meu orientador, o Prof. Osvaldo Pacheco, por ter demonstrado disponibilidade para orientar o meu projecto de mestrado, projecto esse proposto por mim. Agradeço ao meu amigo e ex-colega de curso Hugo Picado por me ter ajudado a rever o documento de dissertação, e pelos conselhos de escrita que me deu. Por fim, mas não menos importante, agradeço a todos os meus amigos e a todas as pessoas que de certa forma marcaram o meu percurso académico e pessoal, que me apoiaram e que têm um papel importante na minha vida.

palavras-chave SASUA, saúde, consulta, registo médico, corpo clínico, utente, calendário

resumo Os SASUA não têm um sistema informático de suporte à gestão da informação clínica dos utentes, de gestão do corpo clínico na perspectiva financeira e de disponibilidade temporal e de gestão das consultas, numa perspectiva integrada. Assim, com este trabalho pretende-se desenvolver um sistema de informação de apoio ao Programa de saúde e bem-estar dos SASUA, que permita a gestão das consultas médicas gratuitas e de todos os dados inerentes, e a gestão dos médicos que exercem essas mesmas consultas. O sistema é direccionado para os utentes do serviço, para os funcionários dos SASUA e para os médicos.

keywords SASUA, health, clinical appointment, medical record, clinical staff, user, calendar

abstract The SASUA don’t have an informatic management system for clinical information of users, medical staff (in the financial perspective and temporal availability) and appointments, in an integrated approach. Thus, this work intends to develop an information system to support the Program of health and welfare of SASUA, allowing to manage all the medical appointments information, calendars and medical staff. The system provides functionalities for users, SASUA’s staff and doctors.

Índice

Lista de Acrónimos

SASUA	Serviços de Ação Social da Universidade de Aveiro
CSU	Centro de Saúde Universitário
LUA	Linha Universidade de Aveiro
UA	Universidade de Aveiro
SNS	Serviço Nacional de Saúde
CE	Comissão Executiva
GAT	Gabinete de Assessoria Técnica
GEPP	Gabinete de Estudos, Planeamento e Prospectiva
UnAF	Unidade Administrativa e Financeira
UnAE	Unidade de Apoio ao Estudante
PACO	Portal Académico Online
SIGEF	Sistema Integrado de Gestão Financeira
SIGAcad	Sistema Integrado de Gestão Académica
SBIDM	Serviços de Biblioteca, Informação Documental e Museologia
SinBAD	Sistema Integrado à Biblioteca Digital
RTS	Rede Telemática Saúde
SISO	Sistema de Informação para a Saúde Oral
PNPSO	Programa Nacional de Promoção da Saúde Oral
ECG	Eletrocardiograma
TCP	Transmission Control Protocol
HTTP	Hyper Text Transfer Protocol
FTP	File Transfer Protocol
JVM	Java Virtual Machine
API	Application Programming Interface
IDE	Integrated Development Environment
UML	Unified Modeling Language
JDBC	Java DataBase Conectivity

SQL	Structured Query Language
PK	Primary Key
FK	Foreign Key
Java EE	Java Enterprise Edition
JPA	Java Persistence API
EJB	Enterprise Java Bean
MBean	Java Managed Bean
JSF	JavaServer Faces
XML	eXtensible Markup Language
HTML	HyperText Markup Language
XHTML	eXtensible HyperText Markup Language
EL	unified Expression Language
URI	Uniform Resource Identifier
CSS	Cascading Style Sheet
SVN	Subversion, Versioning Control Software
SOAP	Simple Object Access Protocol
PGCO	Plataforma de Gestão de Consultas Online
EMR	Electronic Medical Record
EHR	Electronic Health Record
TAC	Tomografia Computorizada
IRM	Imagem de Ressonância Magnética
EUA	Estados Unidos da América

Lista de Figuras e Tabelas

1. Introdução

1.1. Visão Geral

Os Serviços de Acão Social da Universidade de Aveiro (SASUA) integram a Unidade de Apoio ao Estudante que engloba o Núcleo de Saúde. Este núcleo, tem como competência assegurar a oferta de vários serviços médicos, vistos como um apoio complementar ao plano nacional de saúde. As ofertas passam por consultas de várias especialidades completamente gratuitas, a reduções significativas no preço de consultas, exames médicos e internamento em várias instituições externas. Estas vantagens estão ao dispor de alunos, funcionários docentes e não docentes.

Sempre que um utente (aluno ou funcionário) pretende efetuar marcação numa das várias especialidades gratuitas, tem de se dirigir pessoalmente à secretaria dos SASUA ou contactar a mesma por telefone. O processo de marcação de consulta, que demora no máximo cerca de 10 minutos a concluir, pode tornar-se pouco prático e moroso, principalmente em certas alturas do ano em que o tempo de espera para atendimento na secretaria excede uma hora. Relativamente ao acesso às instituições externas, não existe qualquer intervenção por parte dos SASUA, sendo todo o processo tratado exclusivamente entre o utente e a instituição.

Numa universidade onde os estudantes já possuem mecanismos *online* para gerir toda a sua informação académica, a falta de informatização de serviços de uma área importante como a saúde pode ser vista como uma lacuna. Como o Núcleo de Saúde integra o Centro de Saúde Universitário (CSU) que assegura a prática das consultas gratuitas, e este tem vindo a crescer ao longo dos anos, existe ainda mais a necessidade dessa informatização, não só para a gestão de dados administrativos e consultas, mas para toda a manipulação necessária aos dados médicos dos utentes.

1.2. Motivação

Com o intuito de demonstrar melhor o problema dado a conhecer no ponto anterior e que serve de caso de estudo para este trabalho, foi realizado um inquérito junto de um grupo aleatório de pessoas, alunos e funcionários da universidade. Com este inquérito (apresentado no Anexo I), pretendeu-se conhecer melhor os hábitos de utilização do serviço de consultas gratuitas por parte dos utentes, e obter algumas sugestões para o mesmo.

Analisando os resultados obtidos com os 44 inquiridos, foi possível retirar os seguintes valores percentuais (arredondados à unidade):

- 16% já usufruiu do serviço de consultas gratuitas;
- desses 16%, todos efetuaram marcação na secretaria dos SASUA;
- 48% não consideram os métodos de marcação disponíveis práticos e funcionais;

- 80% consideram que seria útil e prático uma plataforma *online* para gestão das consultas.

A adicionar a estes valores, os inquiridos consideram em falta no serviço as especialidades de dermatologia, oftalmologia e medicina dentária.

Os resultados obtidos vêm assim confirmar a falha a nível de sistema informático identificada na análise global do problema. O número reduzido de inquiridos que já usufruiu do serviço de consultas, deveu-se ao facto de muitos dos inquiridos afirmarem desconhecerem a existência do mesmo. Assim, reafirma-se a importância de criar uma plataforma de gestão que possa servir melhor as necessidades dos utentes, vindo colmatar as falhas sentidas na utilização do serviço disponibilizado pelos SASUA.

1.3. Objetivos

O objetivo deste trabalho é desenvolver um sistema *online* de suporte à gestão de consultas. Pretende-se que o sistema possua um conjunto de módulos, que integrados possam satisfazer as necessidades de todas as pessoas envolvidas no Programa de Saúde e Bem-Estar dos SASUA. Essas necessidades passam não só pela gestão de consultas em si, mas também pela gestão de médicos e respectivos calendários, e principalmente pela marcação de consultas por parte dos utentes.

Cada um dos módulos serve um conjunto de utilizadores específico, sendo a divisão feita entre utentes, médicos e funcionários da secretaria dos SASUA. Com esta divisão pretende-se facilitar a gestão dos dados de cada módulo individualmente, embora exista uma comunicação entre os mesmos, para partilhar eventuais informações comuns.

1.4. Estrutura da Dissertação

O documento encontra-se dividido em cinco partes, subdividindo-se cada uma destas em vários pontos. A estrutura usada segue as mesmas linhas de orientação dos documentos de dissertação e tese já apresentados na Universidade de Aveiro.

Este documento é iniciado com uma introdução onde é dada a conhecer uma visão global do problema tratado na dissertação, esclarecendo e enquadrando o leitor sobre os problemas existentes que serviram de motivação à realização do trabalho.

No segundo capítulo pretende-se fazer um enquadramento do caso de estudo da dissertação. Caracterizam-se os Serviços de Acão Social da universidade, fazendo-se referência aos seus objetivos e organização, e dando a conhecer o seu funcionamento. Faz-se também uma breve referência às várias plataformas *online* que a universidade já possui para fins académicos, de cooperação e investigação.

O terceiro capítulo foca-se nas tecnologias de suporte usadas para o desenvolvimento da plataforma pretendida, explicando em que consiste cada uma delas e quais as vantagens e desvantagens do seu uso para o objetivo pretendido.

Nos últimos dois capítulos é apresentado todo o trabalho desenvolvido, desde as ideias base que serviram de alicerces ao projeto, até aos vários passos de desenvolvimento e aos resultados obtidos, terminando com as considerações finais.

2. Enquadramento

2.1. Introdução

Após se ter dado uma visão global sobre o problema em estudo, e antes de se detalhar todo o processo de implementação do sistema pretendido, é importante fazer o enquadramento do problema.

Neste capítulo, será feita uma caracterização dos Serviços de Acão Social da Universidade de Aveiro (SASUA) a nível administrativo geral e a um nível mais específico que incide sobre os serviços de saúde. Considerando que o sistema, apesar de dirigido à comunidade académica em geral, pretende corrigir uma falha existente num núcleo dos SASUA, é importante conhecer certos aspetos do funcionamento desta entidade.

Por fim, será feita uma breve abordagem às plataformas de gestão *online* que a Universidade de Aveiro possui para os mais diversos fins.

2.2. SASUA

2.2.1. Missão

Os SASUA são a entidade da Universidade de Aveiro (UA) responsável por assegurar as funções de ação social necessárias ao ensino superior.

O seu objetivo é proporcionar todos os apoios sociais possíveis aos alunos, permitindo iguais oportunidades no acesso ao ensino superior. Beneficiando os alunos mais carenciados e com aproveitamento escolar, pretende-se que as dificuldades socioeconómicas não sejam motivo de impedimento à frequência de um curso superior [**1**].

2.2.2. Apoios

De entre os apoios concedidos pelos SASUA destacam-se os de apoio direto e indireto. No primeiro grupo através da atribuição de bolsas de estudo, no segundo através do acesso a alimentação, alojamento, serviços de saúde, serviços culturais, desportivos e serviços de biblioteca.

Para além destes apoios referidos, os SASUA oferecem ainda apoios especiais. São estes a atribuição de bolsas de mérito, apoios a alunos com necessidades especiais e sistemas de crédito/empréstimos [**1**].

2.2.3. Organização

Os SASUA são constituídos por quatro órgãos, tendo cada um destes determinadas competências associadas. Os órgãos são: o Conselho de Acão Social, o Administrador para a Acão Social, a Comissão Executiva e o Fórum Social Universitário. É importante referir que apesar de serem parte constituinte da Universidade de Aveiro, os SASUA possuem autonomia administrativa e financeira.

O Conselho de Acão Social, ou CAS, é o órgão superior de gestão e é constituído pelo reitor da UA, pelo administrador dos SASUA e por dois representantes da Associação Académica (sendo um destes, aluno bolseiro). O Administrador para a Acão Social, nomeado pelo reitor da instituição, é legalmente equiparado ao administrador da própria universidade, competindo-lhe toda a responsabilidade administrativa de gestão e execução das políticas de ação social. A Comissão Executiva (CE) é um órgão de apoio técnico ao administrador e é formada pelo próprio juntamente com o diretor da Unidade Administrativa e Financeira, o diretor da Unidade de Apoio ao Estudante, um auditor interno e um representante do Núcleo Jurídico. O Fórum Social Universitário é um órgão exclusivamente consultivo e é formado pelos membros da CE juntamente com um representante de todos os restantes núcleos dos SASUA.

Para garantir a organização dos serviços e das competências necessárias ao funcionamento dos SASUA, estes englobam gabinetes e unidades de apoio que se subdividem em diversos núcleos. Os gabinetes de Assessoria Técnica (GAT) e Estudos, Planeamento e Prospectiva (GEPP) são os gabinetes responsáveis pela parte jurídica, gestão de projetos e obras, comunicação, auditoria, qualidade, entre outros. As unidades Administrativa e Financeira (UnAF) e de Apoio ao Estudante (UnAE) são as responsáveis pela gestão das finanças dos SASUA e por todos os apoios sociais oferecidos aos alunos.

No âmbito deste projeto, a Unidade de Apoio ao Estudante é a que mais relevância tem, englobando os núcleos de Bolsas, Alimentação e Nutricionismo, Alojamento Universitário, Cultura, Desporto e Saúde. O Núcleo de Saúde é responsável por assegurar o funcionamento do Centro de Saúde Universitário (CSU) e do projeto Linha Universidade de Aveiro (LUA), ambos destinados às ofertas de saúde e bem-estar que os SASUA disponibilizam à comunidade académica [**1**].

Para perceber a estrutura organizacional dos SASUA dum modo geral segue-se um organograma legendado com referência a todos os órgãos, gabinetes, unidades e núcleos referidos anteriormente (Fig 1).

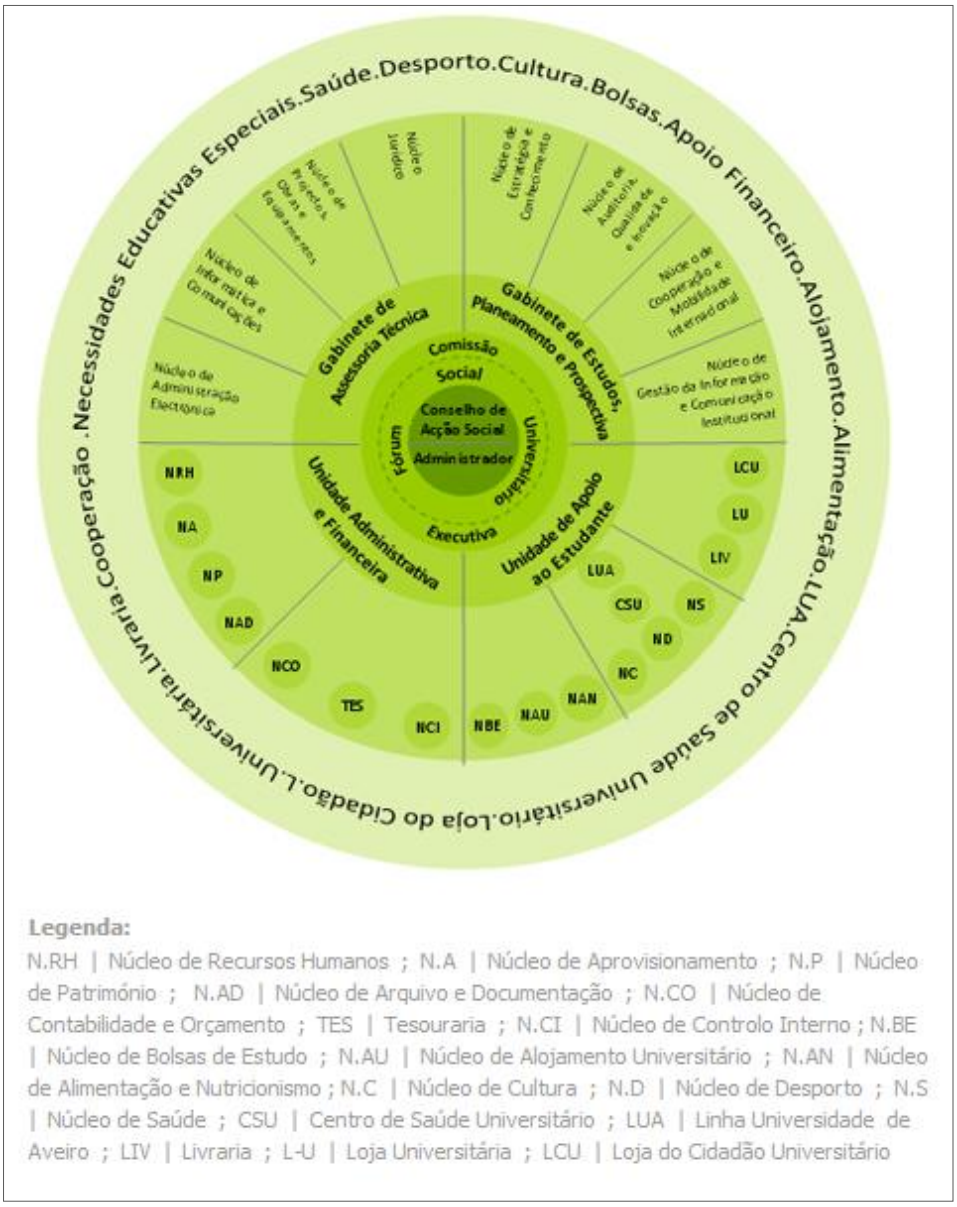


Fig 1: Organograma da estrutura organizacional dos SASUA. [1]

2.2.4. Núcleo de Saúde

Como referido no capítulo inicial, o principal objetivo de todas as ofertas dos SASUA ao nível de saúde é o de complementar os serviços disponíveis pelo Serviço Nacional de Saúde (SNS).

O Núcleo de Saúde tem como competências assegurar a prestação de cuidados de medicina preventiva (consultas, enfermagem, vacinação e campanhas de sensibilização) a toda a comunidade académica, oferecer parcerias com entidades de saúde externas públicas e privadas, promover a

realização de consultas de diversas especialidades e monitorizar todos os dados relativos à utilização do CSU, LUA e das parcerias.

2.2.4.1. Centro de Saúde Universitário

O CSU funciona como um pequeno centro de saúde para a prática de um conjunto de especialidades médicas completamente gratuitas. As consultas (à exceção de uma especialidade) são exercidas num gabinete médico situado no campus universitário, existindo um horário bem definido para cada especialidade. As especialidades disponíveis são:

- Medicina Geral;
- Cirurgia;
- Psiquiatria e Saúde Mental;
- Psicologia;
- Ginecologia e Planeamento Familiar;
- Nutrição e Saúde Alimentar;
- Desabituação Tabágica;
- Medicina do Trabalho.

2.2.4.2. Linha Universidade de Aveiro

O projeto LUA é um projeto inovador e único a nível nacional, baseando-se num serviço de *peer counselling*[1] para estudantes universitários. O objetivo principal é o de oferecer apoio e aconselhamento a universitários que necessitem de ajuda ou se sintam mais sós. É importante referir que uma percentagem relevante de alunos da UA se encontra longe de casa e das respectivas famílias, sofrendo por isso possíveis períodos de adaptação a um meio sociocultural que pode ser bastante distinto do seu.

O LUA existe como serviço na academia desde 1995, sendo na altura uma linha de atendimento noturna para apoiar estudantes universitários através de outros estudantes universitários dispostos a ouvir e ajudar. O serviço funcionava sete dias por semana das 8h da noite às 8h da manhã, sendo tudo tratado de forma completamente confidencial. Em 2008, durante um período experimental, o projeto surgiu em formato virtual através do mundo Second Life. Em 2009 foi dada uma nova ênfase ao projeto com novos meios de contacto e uma maior divulgação [**2**].

Atualmente, o LUA funciona apenas durante o calendário lectivo, de 2ª a 6ª das 9h da noite à 1h da manhã. Apesar do horário de funcionamento mais reduzido, o LUA passou a existir em três formatos:

- **LUA Nightline** – linha telefónica gratuita atendida por alunos com formação específica;
- **e-LUA** – serviço de e-mail destinado a casos menos urgentes mas com a mesma confidencialidade da linha telefónica;

[1] Apoio e aconselhamento dado por pessoas pertencentes ao mesmo ambiente de vivências das pessoas que requisitam a ajuda. Pode ou não efectuar-se de forma anónima.

- **LUA face-to-face** – funcionamento em paralelo com os serviços anteriores para casos que necessitam de ajuda profissional e têm de ser redirecionados para uma consulta de psicologia.

2.2.5. Sistema de Gestão de Consultas

Todos os sistemas de informação e programas informáticos usados pelos SASUA são monitorizados e geridos pelo Núcleo de Informática e Comunicações (parte integrante do GAT). Para além de alguns serviços usados em comum com a UA, como o caso do Sistema Integrado de Gestão Financeira (SIGEF), os SASUA também possuem alguns sistemas próprios como é exemplo o sistema de gestão de consultas.

O sistema de gestão de consultas já existente serve apenas algumas necessidades dos funcionários dos SASUA, não estando disponível para médicos nem para utentes. Este sistema apenas permite a gestão dos médicos que exercem as consultas gratuitas, e a gestão dos respectivos calendários de marcações.

Quando um utente pretende marcar uma consulta, é no sistema existente que a mesma é efetivada. O sistema reconhece automaticamente os dados do utente (nome e número mecanográfico) caso o mesmo já tenha marcado alguma consulta anteriormente, e para cada médico o sistema reconhece se é ou não a 1ª consulta do utente. Contudo, o sistema não tem qualquer mecanismo automático de notificação no caso de desmarcação de consultas, complicando a comunicação entre os SASUA e os utentes quando tal se verifica. Uma outra desvantagem a assinalar, é a falta de integração com os dados de Utilizador Universal e com as respectivas bases de dados de alunos e funcionários.

O facto de esta plataforma não estar disponível para os médicos, obriga a que os mesmos apenas tenham acesso à lista de marcações no próprio dia da consulta, altura em que a lista é divulgada por um funcionário dos SASUA. Se um médico pretender efetuar marcações ou desmarcações de consultas, também não o pode fazer diretamente tendo necessidade de pedir ambas as tarefas a um funcionário.

Em suma, o sistema de gestão existente é muito rudimentar, permite poucas funcionalidades e apenas pode ser acedido por parte de um grupo de utilizadores.

2.3. Plataformas de Gestão Online na UA

A Universidade de Aveiro disponibiliza um grande leque de serviços *online* e sistemas de informação à sua comunidade. O objetivo é de permitir a alunos e funcionários um acesso mais rápido e prático a qualquer informação necessária, e evitar a perda de dados importantes alojando-os em centrais de dados seguras e fiáveis.

As plataformas de gestão consideradas apoiam a comunidade a nível de informação académica, apoio bibliográfico, gestão de materiais, gestão financeira, entre outros.

2.3.1. Apoio Académico

- **MyUA** – serviço que integra *e-mail*, agenda, repositório de dados e outras ferramentas úteis;
- **Moodle** – plataforma de ensino à distância com gestão do material de referência para as cadeiras de todos os cursos;
- **PACO** – plataforma para gestão de dados académicos, inscrições, horários e pedidos de certidões e requerimentos;
- **SIGAcad** – sistema integrado de dados que serve de *backoffice* à plataforma PACO;
- **SBIDM – Catálogo Bibliográfico** – catálogo *online* de livros, revistas, artigos, monografias e a sua disponibilidade na biblioteca da universidade;
- **SinBAD** – sistema integrado com acesso à Biblioteca Digital da UA, disponibiliza artigos audiovisuais, documentos de dissertação e teses.

2.3.2. Investigação

- **SIGIUA** – Sistema Integrado da Gestão da Investigação;
- **RIA** – Repositório Institucional da universidade, armazena, preserva e divulga as produções intelectuais da universidade em formato digital e em regime de acesso livre.

2.3.3. Cooperação

- **Portfólio** – catálogo integrado que disponibiliza todas as valências existentes na instituição (competências e serviços instalados);
- **PmatE** – projeto de investigação e desenvolvimento que desenvolve ferramentas informáticas direcionadas a várias áreas, principalmente à matemática.

2.3.4. Suporte à Atividade

- **Utilizador Universal** – credenciais únicas para o acesso a qualquer serviço online da universidade;
- **E-mail** – serviço de e-mail;
- **Wireless (eduroam)** – rede interna sem fios que cobre todo os campus universitários para uma total mobilidade;
- **WebSig** – sistema completo de gestão de mapas e plantas dos edifícios dos campus universitários que inclui entre outras, referências a lâmpadas e mobiliário;
- **SIGEF** – sistema de suporte para a gestão financeira e contabilística da UA e dos SASUA, inclui planos orçamentais, inventários, contabilidade e gestão de projetos.

2.4. Plataformas Orientadas à Área da Saúde

Nos últimos anos, a Universidade de Aveiro viu vários projetos desenvolvidos por investigadores serem reconhecidos não só a nível nacional como a nível internacional. Alguns desses projetos são direcionados à área da saúde e já se encontram implementados no mercado e em uso por várias instituições externas à UA.

2.4.1. Rede Telemática Saúde

O projeto Rede Telemática para a Saúde (RTS) foi desenvolvido em parceria com a UA no âmbito do Programa Aveiro Digital 2003-2006, um programa com o objetivo de modernizar serviços na zona da Associação de Municípios da Ria. O projeto teve início em 2004, e após uma fase piloto foi oficialmente colocado em funcionamento no dia 5 de Setembro do presente ano.

O objetivo do RTS é o de melhorar e facilitar a troca de informações clínicas entre várias unidades de saúde. Através duma central de dados que aloja o processo clínico dos utentes registados, os médicos de família, enfermeiros e médicos hospitalares podem a qualquer momento ter acesso à informação clínica atualizada do paciente. Esta plataforma, não só melhora a comunicação entre entidades de saúde, como também torna a informação mais segura visto que apenas pessoas autorizadas têm acesso à mesma. Outra vantagem a referir é a redução dos problemas que usualmente ocorrem em sistema de gestão baseados em papel, a perda de informação, o difícil acesso e troca de dados ou a escrita manual por vezes ilegível [**3**].

Dados recentes indicam que a plataforma já registou cerca de 11 milhões de episódios clínicos relacionados com os cerca de 350 mil utentes. As entidades de saúde abrangidas pelo RTS são os hospitais de Aveiro e Águeda, e os Centros de Saúde de Aveiro, Albergaria, Ílhavo, Vagos, Águeda e Oliveira do Bairro [**4**].

2.4.2. SISO

O projeto Sistema de Informação para a Saúde Oral (SISO) foi um projeto desenvolvido pela Universidade de Aveiro em parceria com a Administração Central do Sistema de Saúde (ACSS), a Direção Geral de Saúde (DGS) a Ordem dos Médicos e a Ordem dos Médicos Dentista, em 2007/2008. O objetivo era criar um sistema de informação de apoio ao Programa Nacional de Promoção da Saúde Oral (PNPSO) [**5**].

O SISO pretendia oferecer um pacote de serviços que permitiam a adesão ao PNPSO, a emissão e pagamento de cheques dentista e a prestação de outros cuidados de saúde oral, tudo isto através de uma plataforma de gestão *online*.

O grande objetivo do PNPSO com a criação do SISO, era o de criar uma plataforma para a gestão integrada dos processos inerentes ao Programa do Cheque-Dentista, e facilitar a sua adesão por parte do público-alvo, os idosos e as grávidas acompanhadas pelo SNS. O processo inicia-se com a emissão de um 1º cheque-dentista pelo médico de família. Após esse passo, o utente deve procurar um médico dentista aderente ao programa para usar o cheque em sua posse. Os idosos têm direito a

2 cheques-dentista por ano até finalizar o tratamento necessário, as grávidas têm direito a 3 cheques-dentista por gravidez válidos até 60 dias após o parto [**6**].

2.4.3. VitalJacket

O projeto VitalJacket iniciou-se na Universidade de Aveiro em 2002, na altura sendo apenas um protótipo capaz de medir sinais biológicos como o batimento cardíaco. Atualmente, o projeto é gerido por uma empresa, Biodevices SA., que tem como objetivo o desenvolvimento e comercialização de material biomédico de auxílio ao diagnóstico médico.

O VitalJacket é uma t-shirt que tem acoplada a si uma placa eletrónica capaz de registar um ECG (Eletrocardiograma). O seu uso oferece todo o conforto possível, não tem qualquer cabo ligado a sistemas externos, mantém-se estável ao longo dos movimentos normais do paciente e permite fazer a monitorização durante um longo período de tempo (cerca de 72h).

O acesso aos dados obtidos pode ser feito através do *software* específico do VitalJacket, estando os dados atualizados em tempo real. O transmissor *Bluetooth* (tecnologia wireless) incorporado na placa da t-shirt permite o acesso aos dados remotamente, sendo apenas necessário acesso à internet [**7**].

3. Tecnologias de Suporte

3.1. Introdução

Para desenvolver o projeto e atingir os objetivos pretendidos, foi necessário recorrer a algumas tecnologias de suporte.

Neste capítulo serão apresentadas todas as tecnologias usadas, as linguagens de programação e as ferramentas de desenvolvimento. Será explicado em que consiste cada uma, como se utiliza e qual o seu papel no desenvolvimento do projeto.

O capítulo termina com uma síntese que justifica as opções tecnológicas tomadas em detrimento de outras existentes.

3.2. A Linguagem Java

3.2.1. Características gerais

Java é uma linguagem de programação de alto nível. Foi desenvolvida por James Gosling para a Sun Microsystems (atual propriedade da Oracle Corporation), e apresentada oficialmente em 1995.

A linguagem Java foi criada com base na linguagem C++, sendo ambas orientadas a objetos. O objetivo foi criar algo que fosse relativamente familiar sem as complexidades desnecessárias do C++, permitindo assim aos programadores serem produtivos num curto período de tempo.

Caracteriza-se fundamentalmente por ser uma linguagem simples, dinâmica, robusta, segura e portável. É uma das poucas linguagens orientadas a objetos com capacidade nativa para *multithread*[2]. Foi desenvolvida com algumas regras bem definidas e com o objetivo de necessitar do menor número de dependências possível, tornando-se assim independente de qualquer sistema operativo ou arquitetura. Possui bibliotecas completas para implementação de ligações TCP usando HTTP ou FTP, simplificando imenso o desenvolvimento de código se comparada com o C ou C++ [**8**].

[2] Multithreading no contexto de software é a capacidade de executar threads (fios de execução pertencentes a um mesmo processo) em simultâneo.

3.2.2. Máquina Virtual Java

Quando o código Java é compilado, é produzido um ficheiro com extensão. class por cada ficheiro. java existente. O ficheiro produzido contém o chamado *bytecode* Java. O *bytecode* é o código intermediário usado pelo processador para executar o programa, o mesmo que acontece por exemplo com o C que ao ser compilado produz código *assembly* [**9**].

A Máquina Virtual Java (Java Virtual Machine, JVM) é uma máquina virtual criada para permitir a execução desse *bytecode,* e é o que permite que o mesmo código fonte possa ser executado em qualquer sistema operativo [**10**].

Fig 2: Processo de utilização da JVM. [10**]**

Foram vários os criticismos dirigidos à utilização de uma máquina virtual para execução de código. Ao contrário do que acontece com a execução nativa de código, como por exemplo no C, a utilização de uma máquina virtual juntamente com outras características do Java, tornaram a performance da linguagem pouco aceitável nos primeiros anos. Atualmente, as JVMs existentes estão otimizadas de tal forma que estudos mais recentes demonstram que o Java consegue performances melhores ou similares em comparação com o C, C++ ou C#, dependendo do tipo de operação a efetuar [**11**] [**12**].

3.2.3. Documentação Java

A ferramenta Javadoc foi criada juntamente com o Java para permitir a criação de documentação em formato HTML a partir do código fonte. Todas as APIs (Application Programming Interfaces) e bibliotecas Java possuem o seu código documentado neste formato, e os IDEs (Integrated Development Environments) que suportam Java possuem ferramentas que permitem o acesso a essa documentação em tempo real à medida que o programador desenvolve código [**13**].

Para que seja possível obter a documentação de um determinado ficheiro de código Java, é necessário que os comentários do código se encontrem entre /** e */, existindo algumas *tags* próprias para determinadas finalidades. O exemplo abaixo mostra como documentar em Javadoc um método Java, usando as *tags @param* e *@throws* [**14**].

```
/**
  * Sets the tool tip text.
  *
  * @param text, the text of the tool tip
  * @throws IOException
  */
 public void setToolTipText(String text) throws IOException {}
```

Outras *tags* habitualmente usadas para documentar código são: *@author* (autor), *@version* (versão), *@deprecated* (métodos ou bibliotecas que caíram em desuso ou foram substituídos), @see (referências) e @return (valor ou tipo de retorno).

3.3. Modelação – UML

A linguagem UML (Unified Modeling Language) é um *standard* para modelação de sistemas orientados a objetos no âmbito da engenharia de software. A UML engloba um conjunto de notações para a criação de modelos gráficos ou diagramas. O objetivo destes diagramas é o de criar uma representação gráfica de um projeto desde os seus fundamentos base até fases mais avançadas de desenvolvimento, especificando assim todos os elementos estáticos de um sistema e o seu comportamento à medida que o mesmo evolui [**15**].

A versão estável mais recente de UML (versão 2.3) especifica um conjunto de 14 diagramas, dividindo-se este conjunto entre diagramas estruturais e diagramas comportamentais. Os diagramas estruturais representam os objetos e componentes que vão estar presentes no sistema, é através destes diagramas que se modelam as estruturas de dados, as classes necessárias e respectivos atributos e métodos, e qualquer componente ou serviço externo ao sistema. Os diagramas comportamentais são usados para modelar o comportamento que se espera do sistema, servem para identificar os intervenientes e futuros utilizadores (atores), as ações que os mesmos podem efetuar no sistema, as máquinas de estados que representam os estados possíveis do sistema e qualquer interação entre atores ou ações que se ache relevante [**16**].

No âmbito deste projeto, os diagramas estruturais e comportamentais relevantes são os seguintes:

- Estruturais:
  - diagrama de classes;
  - diagrama de dados (modelo relacional);
  - diagrama de desenvolvimento.
- Comportamentais:
  - diagrama de casos de uso;
  - diagrama de atividade.

Todos os diagramas UML apresentados posteriormente neste documento foram elaborados com a ajuda da ferramenta Office Visio 2007, propriedade da Microsoft. Este *software* foi escolhido

tendo em conta a abundância de elementos UML que possui, para os mais variados tipos de diagramas. O Visio permite também algumas validações automáticas, e permite selecionar o tipo de base de dados a usar (válido em certos tipos de diagramas). Apesar de ser um *software* que exige licença, os alunos da Universidade de Aveiro têm acesso a cópias genuínas do mesmo através da plataforma da Microsoft, MSDN Academic Alliance.

3.4. Base de Dados

3.4.1. Modelo Relacional

O modelo relacional de dados foi criado por Edgar Codd em 1970. O objetivo era encontrar uma estrutura eficaz e lógica para armazenar informação.

Este modelo segue 13 regras de integridade bem definidas. Os dados são representados em forma de tabelas, sendo cada tabela formada por um conjunto de linhas e colunas. Formalmente, os termos tabela, linha e coluna traduzem-se respectivamente em relação, tuplo e atributo. Os nomes dados às colunas de uma tabela devem ser distintos, e pelo menos uma dessas colunas deve obrigar a atribuição de valores não repetidos em cada linha. À coluna com essas propriedades dá-se o nome de chave primária (PK, *Primary Key*) e é o identificador primário de uma tabela [**17**].

Numa base de dados existe frequentemente mais que uma tabela, sendo por vezes necessário que estas se relacionem entre si. O modelo relacional define essas relações como relações binárias, sendo estas divididas nas seguintes nomenclaturas (considerem-se as tabelas A e B):

- 1..1 – relação 1 para 1, um A está associado a um só B, um B está associado a um só A;
- 1..* – relação 1 para vários, um A está associado a um ou mais B, um B está associado a um só A;
- *..* – relação vários para vários, um A está associado a um ou mais B, um B está associado a um ou mais A.

Para que seja possível referenciar uma tabela através de outra quando estas se relacionam, Codd criou o conceito de chave estrangeira (FK, *Foreign Key*). Esta chave serve para identificar o tipo de relação existente, e tal como acontecia com a chave primária, deve garantir-se que nas linhas da tabela o valor da coluna respectiva à chave estrangeira é único. Numa relação do tipo 1..* entre as tabelas A e B, a chave estrangeira em B identifica com qual A esse mesmo B se relaciona [**18**].

3.4.2. Conectividade

Para permitir a conexão entre código fonte e bases de dados, a Sun Microsystems criou uma API conhecida como JDBC (Java DataBase Conectivity). A API JDBC é completamente desenvolvida de

acordo com os *standards* Java e SQL e é suportada pela tecnologia Derby (mecanismos embebidos para bases de dados relacionais) da empresa Apache.

A JDBC permite ao programador conectar-se a uma base de dados e efetuar várias ações sobre a mesma. Adicionar, remover e editar dados, efetuar *queries* (consultas) sobre os mesmos, e processar os resultados obtidos de forma simples, rápida e intuitiva. Todas estas ações são efetuadas através de ferramentas e bibliotecas usadas diretamente no código Java [**19**]. É importante referir que as transações de dados efetuadas com JDBC seguem as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

Atualmente, IDEs como o Netbeans ou o Eclipse possuem ferramentas que permitem a integração com a API JDBC para a criação de bases de dados relacionais sem a necessidade de usar código SQL (como demonstrando na figura Fig 3). A criação de tabelas e respectivas colunas é assim realizada apenas com a definição de alguns campos, sendo o SQL todo gerado automaticamente.

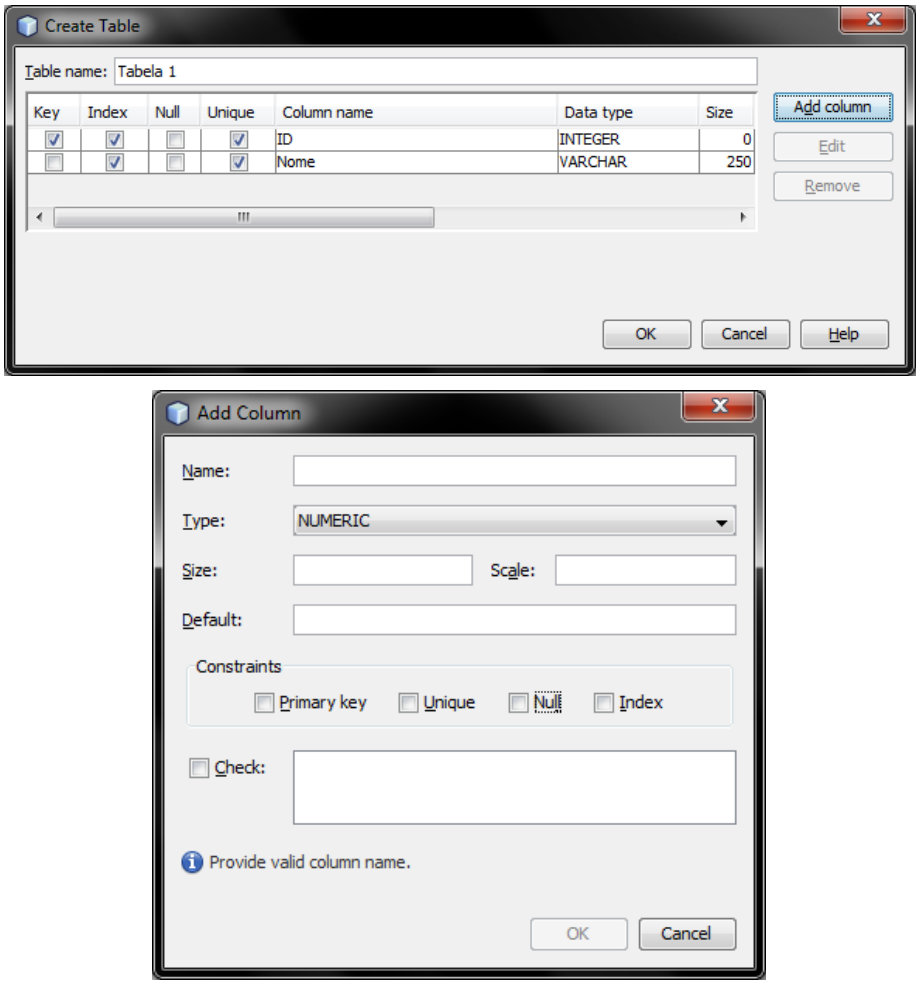


Fig 3: Criação de tabela e respectivas colunas em Netbeans 6.9.1.

3.5. GlassFish

GlassFish, é um *application server* gratuito desenvolvido pela Sun Microsystems para a plataforma Java EE, sendo atualmente propriedade da Oracle Corporation. Um *application server,* pode ser definido neste caso como uma extensão à JVM para lidar com os dois lados existentes, o servidor e o cliente. Oferece ferramentas para a conexão a bases de dados (servidor), bem como para permitir a criação dinâmica das páginas web (cliente). O objetivo é criar um ambiente de execução onde as aplicações possam correr, independentemente do que as mesmas são ou fazem [**20**].

A versão atual do GlassFish já oferece suporte completo para a plataforma Java EE6 e todas as respectivas APIs, sendo apenas concorrido pelo software WebSphere da IBM [**21**] [**22**].

3.6. Java EE – Enterprise Edition

A plataforma Java Enterprise Edition, ou somente Java EE, foi desenvolvida em conformidade com as especificações da linguagem Java, mas com um vasto conjunto de bibliotecas que permitem a criação de aplicações (*server-side*) de forma mais rápida, menos complexa e com uma melhor performance final. Esta tecnologia continua a manter as mesmas linhas de estabilidade e portabilidade verificadas em todas as plataformas Java, encontrando-se atualmente na versão 6 [**23**].

As tecnologias integradas na plataforma Java EE incluem as seguintes APIs (entre outras):

- Managed Beans;
- CDI ou Context and Dependency Injection;
- Persistence e Transaction;
- JavaServer Faces.

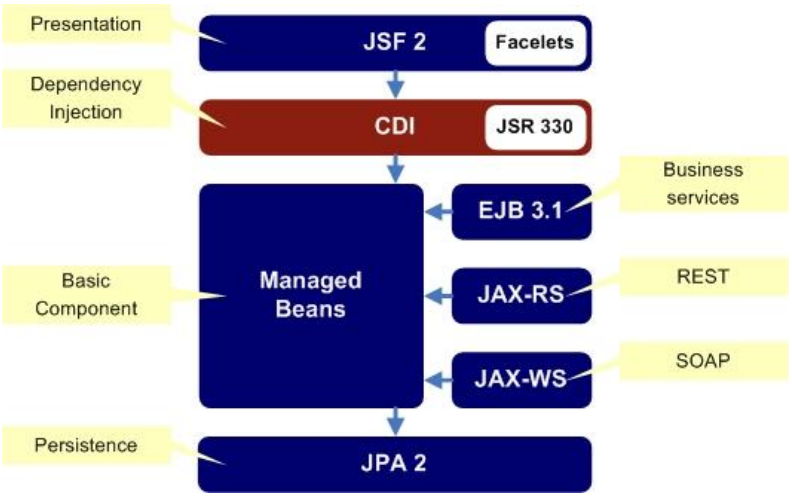


Fig 4: Hierarquia de APIs Java EE6. [20**]**

3.6.1. Persistência

Java Persistence API, ou comummente conhecida apenas por JPA, é a API do Java EE que lida com a forma como uma tabela de uma base de dados relacional é mapeada em objetos Java, ou entidades de persistência. Esta API está inserida nos componentes Enterprise Java Beans (EJBs), que não são mais que componentes do lado servidor que gerem a lógica de negócio de uma aplicação (todos os métodos que servem o propósito da mesma).

A versão atual da JPA vem substituir e melhorar o que já existia numa ferramenta mais simples dos EJBs. O mapeamento da base de dados é feito através da criação de classes entidade que possuem determinadas anotações, essas classes são a ligação direta a cada uma das tabelas, e todos os atributos da classe representam uma coluna das mesmas. Cada classe entidade (ou conjunto de classes) é depois integrada numa unidade de persistência definida através de um ficheiro de configuração próprio. As classes de persistência são geradas automaticamente (usando um IDE) através de cada uma das classes entidade, e englobam todos os métodos necessários às operações de transação efetuadas numa base de dados [**24**].

A classe entidade é indicada através da anotação *@Entity*, seguindo-se a anotação *@Table* que indica o nome da tabela mapeada, o *schema* e as colunas da tabela que não aceitam dados repetidos (referenciados através da anotação *@UniqueConstraint*). Entre a anotação *@Entity* e a declaração da classe podem ser definidas várias *queries* a aplicar na tabela correspondente.

A definição dos atributos da classe entidade também está sujeita à utilização de algumas anotações. As anotações *@Basic* e *@Column* são obrigatórias a todos os atributos, sendo a anotação *@Id* usada apenas no atributo definido como chave primária da tabela. Se existir alguma relação com outra tabela da mesma base de dados, esta também deve ser definida na classe entidade através da anotação correspondente ao tipo de relação entre as tabelas, por exemplo, *@OneToOne* (1..1).

No contexto das classes entidade, as anotações a reter são as seguintes:

- **@Entity** – declaração de classe entidade;
- **@Table** – declaração dos dados da tabela;
- **@UniqueConstraint** – declaração dos nomes das colunas que não suportam repetição de dados;
- **@Id** – definição do atributo correspondente à chave primária;
- **@GeneratedValue** – definição do modo como os valores da chave primária são gerados, por defeito o valor é gerado automaticamente;
- **@Basic** – definição da obrigatoriedade do atributo aquando da adição de uma nova entrada na tabela;
- **@Column** – declaração do nome da coluna correspondente ao atributo da classe;
- **@JoinColumn** – declaração da coluna usada como chave estrangeira numa relação entre tabelas;
- **@OneToOne** – declaração de uma relação do tipo 1..1;
- **@OneToMany** – declaração de uma relação do tipo 1..*;
- **@ManyToOne** – declaração de uma relação do tipo *..1, usa-se em conjunto com a anotação anterior, cada uma na respectiva classe entidade;
- **@ManyToMany** – declaração de uma relação do tipo *..*.

Na figura seguinte (Fig 5), encontra-se exemplificado o uso de algumas anotações numa classe entidade.

```
@Entity
@Table(name = "UTENTE", catalog = "", schema = "APP", uniqueConstraints = {
    @UniqueConstraint(columnNames = {"MEC"}),
    @UniqueConstraint(columnNames = {"LOGIN"}),
    @UniqueConstraint(columnNames = {"NOME"})})

    /**/

public class Utente implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Basic(optional = false)
    @Column(name = "IDUTENTE", nullable = false)
    private Integer idutente;
    @Basic(optional = false)
    @Column(name = "LOGIN", nullable = false, length = 100)
    private String login;
    @Basic(optional = false)
    @Column(name = "PASSWORD", nullable = false, length = 100)
    private String password;

    /**/

    @Basic(optional = false)
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "utente")
    private Calendario calendario;

    public Utente() {
    }
```

Fig 5: Classe entidade 'Utente' que mapeia a respectiva tabela 'Utente'.

Na classe de persistência gerada automaticamente através da classe entidade, são definidos os métodos necessários às operações atómicas duma base de dados: a criação, edição e remoção de uma linha (tuplo) da tabela. Para além destes métodos, a classe disponibiliza também ao programador vários métodos de procura.

Todas as operações são tratadas através de um Entity Manager. Este é definido através duma unidade de persistência (no caso da Fig 6, a PGCO_UtentePU) que gere todas as entidades de uma mesma base de dados. Estas definições são configuradas através de um ficheiro XML normalmente nomeado de persistence.xml. É também neste ficheiro de configuração que se define qual a base de dados associada às classes da unidade de persistência, ou seja, a base de dados que contém as tabelas através das quais as classes entidade foram geradas.

```
public class UtenteJpaController {

    public UtenteJpaController() {
        emf = Persistence.createEntityManagerFactory("PGCO_UtentePU");
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(Utente utente) throws PreexistingEntityException, Exception
    {...}

    public void edit(Utente utente) throws IllegalOrphanException, NonexistentEntityException, Exception
    {...}

    public void destroy(Integer id) throws IllegalOrphanException, NonexistentEntityException
    {...}

    public List<Utente> findUtenteEntities()
    {...}
```

Fig 6: Classe de persistência da entidade 'Utente'.

Toda esta geração automática de código através das tabelas de uma base de dados é uma grande ajuda no desenvolvimento de um projeto. O programador não tem necessidade de escrever todo o código manualmente, reduzindo assim o tempo de desenvolvimento. Contudo, também existem desvantagens inerentes ao uso destas ferramentas. Quando é necessário alterar o nome ou o tipo de uma coluna de uma tabela, ou adicionar uma nova coluna, é necessário atualizar a classe entidade correspondente à tabela alterada, e isto pode implicar alterar também outras classes entidade (na existência de relações). Após a alteração das classes entidade necessárias, é também importante que se atualizem as respectivas classes de persistência, sendo após isso feita uma verificação ao código que usa qualquer uma das classes alteradas. Em suma, uma pequena alteração na base de dados gera toda uma "bola de neve" de alterações em vários pedaços de código, o que numa fase mais avançada dum projeto pode resultar em várias complicações de integridade.

3.6.2. Managed Beans

Os Java Managed Beans são objetos que representam recursos (aplicações, serviços ou componentes). O seu uso está diretamente associado ao uso da API JavaServer Faces, como será explicado no ponto seguinte (3.6.3), contudo, é importante esclarecer como estes objetos são usados numa aplicação web.

Um MBean, como é comummente conhecido, assemelha-se em tudo a uma classe Java seguindo as mesmas regras de boa programação em linguagens orientadas por objetos. A classe deve ter um construtor por omissão (sem argumentos) e todos os atributos devem ser privados, sendo acedidos através dos respectivos métodos *set* e *get*. A estas características é adicionado um conjunto

de outras especiais que tornam os MBeans parte de uma ferramenta poderosa dentro da tecnologia Java EE6.

A declaração que define uma classe como um MBean pode ser feita de duas formas: através de uma anotação ou num ficheiro de configuração em formato XML. A anotação *@ManagedBean* antes da declaração da classe foi adicionada na versão atual da API juntamente com anotações para definir o *scope* do Bean. A inserção destas anotações teve como objetivo evitar o uso do ficheiro de configuração obrigatório na versão anterior, o faces-config.xml [**25**]. Usar apenas o ficheiro XML, permite contudo ter as configurações juntas num mesmo ficheiro e de mais fácil interpretação, podendo-se obter esse resultado diretamente no IDE aquando da criação do MBean.

Os *scopes* de um MBean podem ser quatro e definem o tempo de vida que o mesmo tem numa aplicação e quais os utilizadores que podem aceder às suas instâncias. Os *scopes* disponíveis são os seguintes:

- **@RequestScoped** – usado por omissão; é criada uma nova instância da classe sempre que existir um HTTP Request;
- **@ViewScoped** – é usada a mesma instância da classe para o mesmo utilizador numa mesma página;
- **@SessionScoped** – é usada a mesma instância da classe até a sessão do utilizador ser terminada pelo próprio ou por um *timeout*; usada preferencialmente para MBeans que implementam métodos de login;
- **@ApplicationScoped** – a instância da classe é partilhada por todos os utilizadores; existem vários pontos a ter em conta aquando da utilização deste *scope* por uma questão de privacidade, segurança e integridade de dados.

Existe ainda um *@NoneScoped*, apesar de não ser muito frequente o seu uso. O objetivo é não definir um *scope* para o MBean por este ser apenas um auxiliar usado por outro MBean, tendo já este último de ter um *scope* definido [**26**].

3.6.3. JavaServer Faces

A API JavaServer Faces, ou somente JSF, é uma *framework* para a criação de interfaces de utilizador em aplicações Java EE6. Foi desenvolvida com o objetivo de simplificar o acesso e a gestão dos dados do lado servidor através da interface para o cliente, disponibilizando um conjunto de *standards*, componentes HTML e ferramentas para validação de dados.

Das suas principais características, destacam-se como relevantes para o desenvolvimento deste projeto o conjunto completo de componentes web e HTML, a interação direta com MBeans, e o acesso aos atributos e métodos dos mesmos permitida pela integração com a linguagem EL (unified Expression Language), uma linguagem maioritariamente usada em programas Java que permite usar expressões embebidas nas páginas web.

A criação da interface web inicia-se através dum documento base XHTML. Os componentes, incluindo <body> e <head>, passam a ser acedidos através do prefixo ‘h’ correspondente aos elementos da biblioteca HTML, usando-se em situações mais específicas o prefixo ‘f’ que corresponde aos elementos da biblioteca central do JSF [**27**]. Os elementos com estes prefixos são disponibilizados

através da adição de dois URIs (Uniform Resource Identifiers) à *tag* <html> do ficheiro web, como exemplificado de seguida.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
       (…)
```

No cabeçalho (<head>) do ficheiro XHTML é também identificado o URI para o ficheiro de estilos. Neste ficheiro, que segue as diretrizes da linguagem CSS (Cascading Style Sheet), são definidas várias propriedades relacionadas com o aspeto e formatação das páginas web, tamanhos, tipos de letra, cores, espaçamentos, limitações, menus, entre outras.

Neste contexto, os MBeans são conhecidos como Page Beans ou Backing Beans, pois são os responsáveis por suportar os atributos e métodos necessários ao funcionamento das interfaces no seu ciclo de vida. São estes que recebem os dados submetidos pelo utilizador ao efetuar uma ação, e depois de devidamente tratados, são os responsáveis por enviar a informação de retorno necessária à atualização da interface.

Para aceder a um MBean e poder efetuar trocas de informação entre cliente e servidor, são usados os atributos *binding*, *action* e *value* dos elementos do ficheiro XHTML. Quando a aplicação é executada e a página é processada, o MBean é instanciado automaticamente (daqui advém a necessidade de um construtor sem argumentos) e os seus atributos são acedidos através dos respectivos métodos *set* e *get*. Este acesso é efetuado através da expressão "#{MBean.info}", em que 'MBean' assume o nome do Managed Bean e 'info' assume o nome do atributo ou método necessário [**28**].

```
<h:form id="login" styleClass="form_settings">
    <label>Utilizador Universal</label>
    <h:inputText id="userIt" styleClass="input" size="30" value="#{Session.user}"/>
    <h:message style="color: red" for="userIt"/>
    <label>Password</label>
    <h:inputSecret styleClass="input" size="30" value="#{Session.password}"/>
    <p style="padding-top: 8px;">
        <h:commandButton styleClass="submit" value="OK" action="#{Session.login}"/>
    </p>
</h:form>
```

Fig 7: Exemplo da utilização das tags e atributos JSF em XHTML.

3.6.4. Regras de Navegação

Como acontece em qualquer aplicação web, as ações do utilizador podem originar a atualização ou mudança de página. Para que isto se processe, é necessário ter definido anteriormente um conjunto de regras conhecidas como regras de navegação.

Na tecnologia Java EE6, estas regras de navegação são definidas num ficheiro XML denominado de faces-config.xml. Para cada ação numa determinada página, é necessário definir se essa mesma página é atualizada ou se o utilizador deve ser redirecionado para uma nova. As decisões de navegação têm como critério base o valor de retorno do método associado à ação realizada pelo utilizador. Estas decisões podem ser definidas de acordo com regras estáticas, dinâmicas ou através de condições (*if … then)* [**29**].

Tal como referido anteriormente, este ficheiro de configuração pode também conter a definição dos MBeans, o nome pelo qual os mesmos são referenciados nas páginas XHTML e o seu *scope* (Fig 8).

```
<navigation-rule>
   <from-view-id>from_page.xhtml</from-view-id>
   <navigation-case>
      <from-action>#{ManagedBean.actionMethod}</from-action>
      <from-outcome>condition 1</from-outcome>
      <to-view-id>/to_page1.xhtml</to-view-id>
   </navigation-case>
   <navigation-case>
      <from-action>#{ManagedBean.actionMethod}</from-action>
      <from-outcome>condition 2</from-outcome>
      <to-view-id>/to_page2.xhtml</to-view-id>
   </navigation-case>
</navigation-rule>
```

Fig 8: Exemplo de um ficheiro faces-config.xml. [29**]**

3.7. Google Calendar

O número de APIs Google tem vindo a crescer drasticamente nos últimos anos, e consequentemente também o seu uso em aplicações web. As APIs são todas gratuitas, fornecem tutoriais de utilização e possuem documentação em formato Javadoc para as bibliotecas necessárias. Desde mapas, gráficos, calendários até ferramentas para a criação de sites, as APIs Google fazem da empresa a mais completa na oferta deste tipo de serviços gratuitos [**30**].

A aplicação de calendário da Google (Google Calendar) permite ao utilizador criar vários calendários numa mesma conta e geri-los como uma agenda de eventos, disponibilizando a informação à distância de um clique com um simples acesso à internet. Esta aplicação não só pode ser usada num browser acedendo diretamente através dos *microsites* do Google, como também pode

ser embebida numa qualquer aplicação web. A Universidade de Aveiro usa os serviços Google Calendar embebidos no *site* do projeto LUA (Linha Universidade de Aveiro, 2.2.4.2), dando a conhecer publicamente a escala de serviço dos voluntários do projeto.

O uso da API Calendar em formato embebido não disponibiliza ao utilizador todas as ações permitidas através da aplicação online, excluindo, por exemplo, a capacidade de registar novos eventos ou criar novos calendários. Contudo a aplicação embebida pode ser muito útil e prática para a leitura e sincronização de informação. São estas algumas das vantagens exploradas no projeto, permitindo a um utilizador do módulo de utentes sincronizar o seu calendário de consultas com um calendário na sua conta Google, assim, sem necessitar de aceder à plataforma de gestão de consultas o utilizador tem disponível toda a sua agenda.

Apesar da utilização deste serviço obrigar os utilizadores a possuir uma conta Google, o mesmo traz mais vantagens daquelas já mencionadas. Um calendário Google pode atualmente ser utilizado em várias outras aplicações, por exemplo, na aplicação Mozilla Thunderbird através da extensão Lightning, ou mesmo com o Outlook da Microsoft.

Para que seja possível embeber um calendário Google numa aplicação, é necessário usar o endereço privado do utilizador para esse mesmo calendário. Através desse endereço a aplicação retorna um *feed* em formato XML com todas as informações que o calendário possui. Para tornar possível este serviço através de XML, é usado o protocolo SOAP (Simple Object Access Protocol), um protocolo de comunicação que define o formato de mensagens XML, permite a sua construção e o seu processamento [**31**] [**32**].

3.8. Repositório – Subversion

Quando se desenvolve um projeto que pode sofrer várias alterações ao longo do tempo, e quando pode ser crucial ter acesso a versões anteriores do projeto, é importante ter uma forma segura e organizada de o armazenar. Com esse objetivo surgiram os repositórios de dados.

A ferramenta Subversion, ou simplesmente SVN, é uma ferramenta que permite aceder a código ou documentação através do seu histórico de versões. Cada vez que a alteração num determinado ficheiro é guardada, é definida uma nova versão do mesmo, sendo possível aceder a todas as versões desde a data de criação. O SVN permite também a comparação de ficheiros, vendo assim quais as alterações efetuadas de uma versão para outra. Ainda que não tenha sido necessário neste projeto, a ferramenta SVN permite trabalho colaborativo, ou seja, os ficheiros do repositório podem ser acedidos e modificados por várias pessoas, permitindo no final saber quem efetuou que alterações e dando a conhecer quais as estatísticas de trabalho de cada elemento.

Os IDEs Netbeans e Eclipse permitem completa integração com ferramentas de revisão e versão como o SVN. O utilizador pode através do IDE fazer o *download* ou *upload* de versões de ficheiros do repositório, sendo-lhe permitido também reverter modificações e substituir um determinado ficheiro por uma versão anterior.

3.9. Porquê Java EE6?

As discussões sobre as diversas linguagens de programação e respectivas plataformas de desenvolvimento, levam sempre à mesma conclusão, não existe uma melhor que todas as outras. Os fatores a ter em consideração são demasiados, todas têm as suas vantagens e desvantagens, contudo, ao começar um projeto de programação é efetivamente necessário fazer uma escolha.

Optar por usar a tecnologia Java EE6 em detrimento de outra, teve como principais razões o nível de conhecimentos já obtidos anteriormente, o à vontade com a tecnologia, as vantagens inerentes da mesma e a quantidade de recursos gratuitos disponíveis.

A minha experiência pessoal com Java é um ponto importante a ter em consideração. Iniciar um projeto, ainda que de pequena ou média dimensão, e começar por ter de obter conhecimentos razoáveis sobre uma nova linguagem ou plataforma diminui o nível de produtividade inicial e atrasa todo o projeto quando se tem uma *deadline* fixa. O à vontade com determinada tecnologia facilita também a aprendizagem na utilização de novas APIs associadas que possam vir a ser necessárias.

O desenvolvimento de aplicações web tem atualmente como base duas principais plataformas: Java EE6 da Oracle Corporation e C#.NET da Microsoft. Naturalmente, e como já referido no início deste ponto, as críticas dividem-se, contudo, e como referência de popularidade, segundo o índice TIOBE[3] a linguagem Java é atualmente a mais popular e mantém esta posição desde 2006 [**33**].

Optar por Java EE6 em detrimento de .NET tem algumas vantagens importantes, que podem fazer toda a diferença. Java sempre foi portável e multiplataforma, ao contrário de .NET que apenas recentemente ganhou essa qualidade e apenas através de uma plataforma de auxílio para Linux e Mac OS conhecida por Mono [**34**]. Java é livre, as APIs são livres, os IDEs com os quais se pode desenvolver código Java são livres. A evolução da linguagem e das várias APIs através das versões lançadas, permite o desenvolvimento de aplicações robustas e seguras sem preocupações relativamente a compatibilidades. No caso da oferta da Microsoft o cenário é o oposto. A plataforma não é livre, o melhor IDE de desenvolvimento também requer uma licença e são muitos os problemas conhecidos a nível de compatibilidade entre versões [**35**].

Para finalizar, e como aspeto importante no desenvolvimento de aplicações integradas com vários serviços externos, estes encontram-se disponíveis livremente em muito maior número para aplicações Java. Por exemplo, todos os serviços Google (Health, Calendar, ToolKit, entre outros) encontram-se disponíveis com documentação apropriada para integrar em qualquer aplicação web Java.

[3] Índice de popularidade das linguagens de programação. É baseado nas pesquisas realizadas no Google, Yahoo!, MSN, Wikipedia, entre outros, e actualizado mensalmente.

4. Plataforma de Gestão de Consultas Online – PGCO

4.1. Introdução

O desenvolvimento de um projeto atravessa várias etapas. Começando pelas ideias base que serviram de motivação e alicerces ao projeto, até ao produto final, este capítulo documenta todas as etapas ultrapassadas e o que resultou de cada uma dessas etapas.

O capítulo inicia-se com a obtenção de requisitos para o sistema, a identificação dos atores envolvidos e as ações relacionadas com cada um (funcionalidades). Num segundo ponto será documentada a modelação do sistema com a ajuda de diagramas UML, identificando-se os dados necessários a cada módulo e respectivas restrições, expondo graficamente as conclusões do ponto anterior. Seguidamente à modelação será feita referência às interfaces de utilizador e às linhas de orientação seguidas para a sua construção.

Para terminar o capítulo, e mais uma vez recorrendo a diagramas UML, serão explicadas detalhadamente todas as funcionalidades de cada módulo.

4.2. Análise de Requisitos

O ponto de partida de qualquer projeto de *software* deve ser a análise de requisitos. Esta análise consiste em identificar o conjunto de utilizadores a quem o sistema se destina (atores), e em identificar todas as ações que cada ator vai poder realizar no sistema.

Ainda que um projeto possa sofrer alterações ao longo do seu desenvolvimento, uma boa análise de requisitos inicial garante à partida uma base mais sólida e estruturada, evitando assim problemas maiores na realização do projeto.

4.2.1. Utilizadores

A Plataforma de Gestão de Consultas Online (PGCO) é destinada a três grupos de utilizadores. Os utentes do serviço (alunos e funcionários docentes e não docentes), os funcionários da secretaria dos SASUA e os médicos que exercem as consultas gratuitas perfazem o conjunto de utilizadores futuros da plataforma. Assim, identificam-se os atores Utente, Funcionário SASUA e Médico.

4.2.2. Funcionalidades a Suportar

Após a identificação dos atores do sistema, é necessário identificar as necessidades de cada ator e quais as funcionalidades a implementar para satisfazer essas necessidades.

A plataforma deve estar protegida através dum acesso condicionado, permitindo apenas o seu uso aos utilizadores referidos anteriormente. A forma mais eficaz de condicionar o acesso à plataforma é através dum sistema de login. Identifica-se assim a primeira funcionalidade, comum a todos os atores.

Deve ser permitido aos utentes e médicos a gestão completa dos seus calendários de marcações. Para satisfazer estas necessidades, a plataforma deve possuir as funcionalidades de marcação e desmarcação de consultas, visualização do calendário de marcações e visualização dos dados do utente com consulta marcada (esta última funcionalidade aplica-se apenas aos médicos). Para os funcionários dos SASUA, e de modo a oferecer a maior confidencialidade possível aos utentes, as funcionalidades a implementar passam apenas pela desmarcação de consultas (efetuadas a pedido de um médico) e pela visualização dos *slots*[4] temporários livres e ocupados para cada médico num determinado dia (estes utilizadores não têm acesso ao nome dos utentes com consultas marcadas).

O registo de novos utilizadores na plataforma é também uma funcionalidade a considerar. Os utentes podem registar-se quando efetuam o primeiro acesso à plataforma, os médicos devem ser registados por funcionários dos SASUA, sendo estes utilizadores os únicos com permissões para efetuar posteriores alterações ao registo de um médico.

Em suma, as funcionalidades gerais que a plataforma deve suportar identificam-se na tabela seguinte:

	Utentes	Funcionários SASUA	Médicos
login por Utilizador Universal	✓	✓	✓
informação em tempo real	✓	✓	✓
registo de médicos	×	✓	×
registo de utentes	✓	×	×
dados do utente	✓	×	✓
marcação de consultas	✓	×	✓
desmarcação de consultas	✓	✓	✓
calendário de marcações	✓	×	✓
slots disponíveis	✓	✓	✓

Tabela 1: Sumário de funcionalidades a suportar.

[4] Um slot temporário, é neste contexto um espaço temporal reservado à realização de uma consulta. Este tempo pode variar entre especialidades.

4.3. Modelação do Sistema

4.3.1. Visão Geral

Depois de identificados os atores e as funcionalidades para a plataforma, é necessário modelar o sistema. A modelação consiste em identificar todas as entidades participantes e a forma como as mesmas interagem entre si. No final, obtém-se um plano de desenvolvimento que inclui todos os módulos envolvidos, as bases de dados e os serviços externos, bem como as ligações entre todas estas entidades (Fig 9).

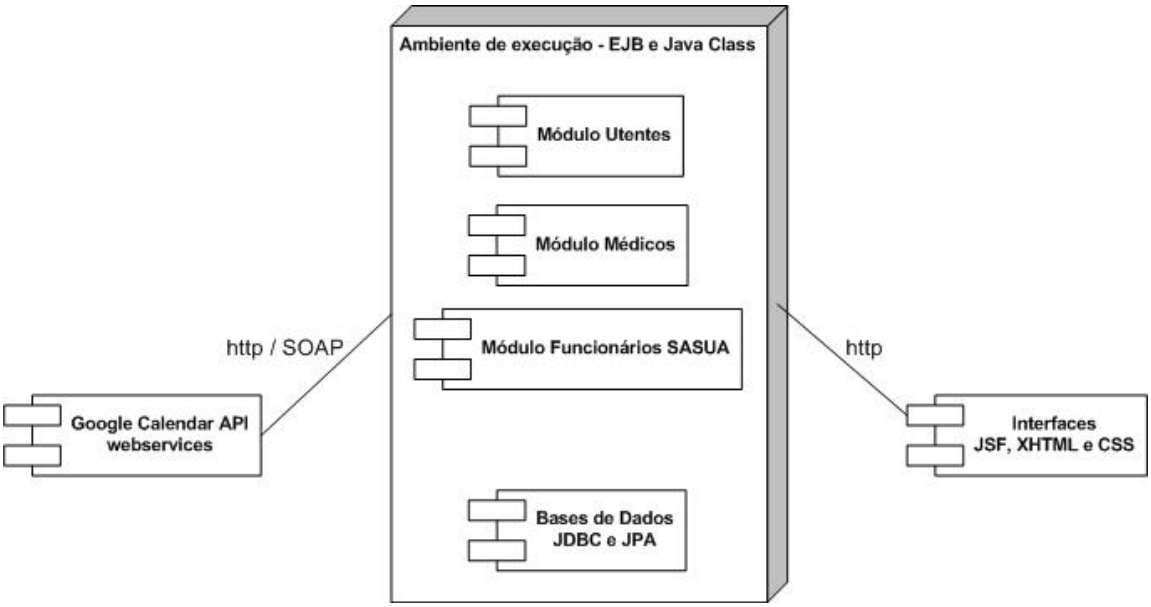


Fig 9: Diagrama de desenvolvimento para a implementação da plataforma.

A plataforma é subdividida em três módulos, um para cada grupo específico de utilizadores. Apesar da divisão estabelecida, existe informação comum sobre os utentes e consultas, sendo necessário assegurar a comunicação entre cada um dos módulos.

Toda a informação a partilhar internamente está contida nas bases de dados, uma para cada módulo. A comunicação do utilizador com o módulo que se lhe destina é realizada através da interface web. É importante referir que para o desenvolvimento do projeto foi ainda usada uma base de dados auxiliar, o objetivo desta é simular a base de dados da Universidade de Aveiro onde estão alojados os dados administrativos dos seus membros. Assim, automaticamente o acesso à plataforma fica limitado apenas aos utilizadores com credenciais de Utilizador Universal.

Externamente e para implementação das funcionalidades de calendário, a plataforma comunica com a API Google Calendar. Todo o conteúdo dos calendários é armazenado nas bases de dados do serviço do Google.

4.3.2. Casos de Uso

Os casos de uso refletem as ações permitidas a cada ator num determinado sistema. Para os atores identificados para a PGCO (Utente, Funcionário SASUA e Médico) foram elaborados os respectivos diagramas UML de casos de uso.

O Utente pode efetuar login, registar-se na plataforma, ver e editar os seus dados pessoais. Relativamente às consultas médicas, o Utente pode marcar e desmarcar consultas, ver o calendário de marcações e sincronizar esse mesmo calendário com um calendário Google (Fig 10).

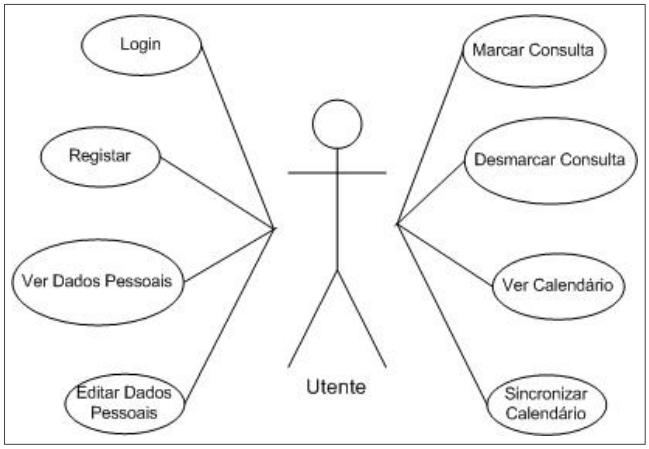


Fig 10: Diagrama de casos de uso - Utente.

O Funcionário dos SASUA pode efetuar login e ver os seus dados pessoais. É o ator a quem estão associadas as tarefas associadas aos registos médicos, registar um novo médico, ver os dados do médico, editar ou remover os dados de um médico registado. O Funcionário dos SASUA pode também desmarcar consultas, enviar notificações e ver os *slots* disponíveis para cada médico (Fig 11).

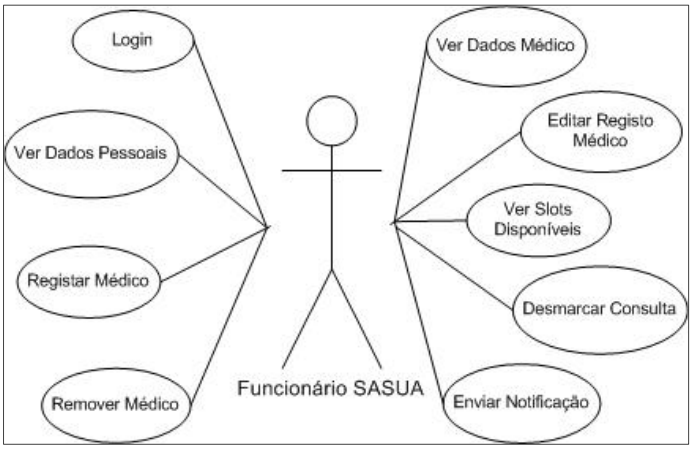


Fig 11: Diagrama de casos de uso - Funcionário SASUA.

O Médico pode efetuar login e ver os seus dados pessoais. Pode também marcar e desmarcar consultas, ver o calendário de marcações e os dados dos utentes com consultas marcadas, e enviar notificações (Fig 12).

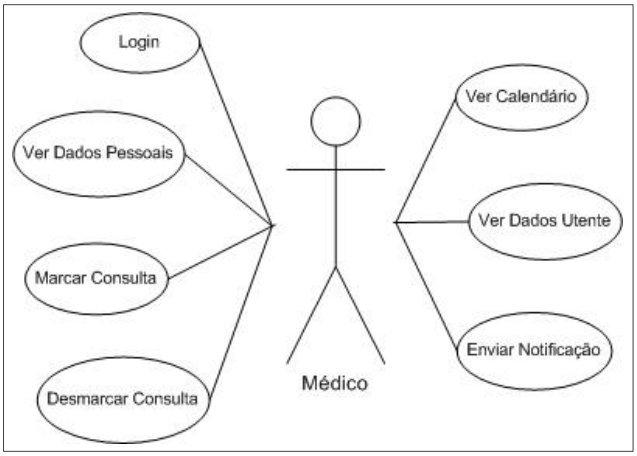


Fig 12: Diagrama de casos de uso - Médico.

4.3.3. Dados e Relações

O passo seguinte após identificados os atores do sistema e todas as funcionalidades inerentes, é identificar os dados associados a cada ator e consequentemente ao módulo respectivo. Estes dados vão caracterizar as várias classes ou entidades envolvidas, e vão posteriormente formar as colunas de cada tabela da base de dados associada à PGCO.

A modelação dos dados associados a cada classe identificada é realizada através de diagramas de classe, estes representam o nome da classe, os respectivos atributos e o seu tipo. Com estes diagramas representam-se também as relações que associam as várias tabelas (de acordo com o modelo relacional).

Como referido no ponto 4.3.1, foi criada uma base de dados para simular os dados administrativos dos utentes já existentes nas bases de dados da Universidade de Aveiro. O objetivo é simular a obtenção automática dos dados aquando do registo de um utente, evitando assim que o mesmo tenha de preencher todos os campos necessários. De notar que alguns campos não podem sequer ser alterados (nome, número mecanográfico, login universal, data de nascimento e sexo são dados que não variam), e o preenchimento automático do registo evita assim possíveis erros introduzidos pelo utilizador.

A classe UUniversal modela os dados associados à base de dados simulada, englobando todos os dados administrativos de um utilizador necessários ao funcionamento da plataforma.

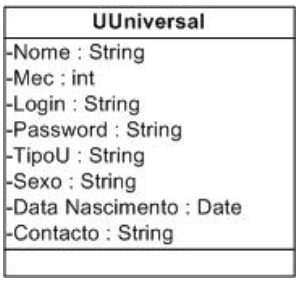


Fig 13: Diagrama de classes - Utilizador Universal.

Para o módulo associado aos utentes, identificam-se três classes: Utente, CalendarioUtente e ConsultaUtente. A classe Utente engloba os dados administrativos associados aos utentes do serviço e os necessários para implementar as funcionalidades direcionadas a estes utilizadores. Os atributos Login, Nome, Mec (número mecanográfico) e Contacto são únicos a cada utente. O atributo TipoU identifica o tipo de utente em questão, aluno ou funcionário. A classe CalendarioUtente tem um só atributo, uma expressão que identifica o calendário e que consiste no login do utilizador. A classe ConsultaUtente é caracterizada pelos dados normais de uma consulta, a data e hora de ocorrência, o médico e a respectiva especialidade.

A classe Utente relaciona-se com a CalendarioUtente com uma relação do tipo 1..1, ou seja, um utente está associado a um único calendário e vice-versa. A classe CalendarioUtente possui uma relação do tipo 1..* com a ConsultaUtente, significando isto que o calendário de um utente pode estar associado a várias consultas, mas que as consultas de um utente estão associadas apenas ao seu calendário.

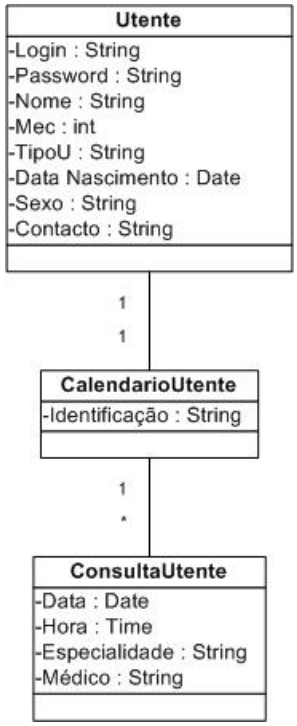


Fig 14: Diagrama de classes - Utente.

No módulo associado aos funcionários dos SASUA é necessária apenas uma classe, denominada de FuncSASUA. Todas as funcionalidades de edição de dados associadas a estes utilizadores implicam a alteração de informação médica, assim, faz sentido que os dados sejam manipulados diretamente nas tabelas associadas ao módulo dos médicos. A classe FuncSASUA engloba os dados administrativos associados ao funcionário, sendo os atributos Login e Nome únicos a cada utilizador.

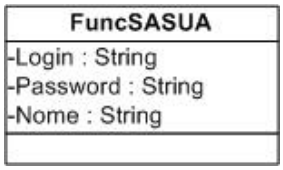


Fig 15: Diagrama de classes - Funcionário SASUA.

As três classes identificadas para o módulo associado aos médicos são: Medico, CalendarioMedico e ConsultaMedico. A classe Medico engloba os dados administrativos do médico e os dados associados ao seu horário de consulta. Os atributos Login e Nome são únicos a cada médico. Os atributos DiaConsulta e DiaConsultaInt identificam, respectivamente, o dia da semana em que as consultas são exercidas e o dia em valor numérico (1 corresponde a 2ª feira). Os atributos IniSlotConsultas e FimSlotConsultas correspondem à hora a que as consultas do médico começam e a hora a que terminam, e o atributo NumConsultas corresponde ao número máximo de consultas exercidas pelo médico em cada dia. A classe CalendarioMedico tem um só atributo, uma expressão que identifica o calendário e é formada pelos nomes da especialidade e do médico. A classe ConsultaMedico é caracterizada pela data e hora de ocorrência, e pelo nome do utente que tem a consulta marcada nessa mesma data e hora.

A classe Medico relaciona-se com a CalendarioMedico com uma relação do tipo 1..1, ou seja, um médico está associado a um único calendário e vice-versa. A classe CalendarioMedico possui uma relação do tipo 1..* com a ConsultaMedico, significando isto que o calendário de um médico pode estar associado a várias consultas, mas que as consultas de um médico estão associadas apenas ao seu calendário.

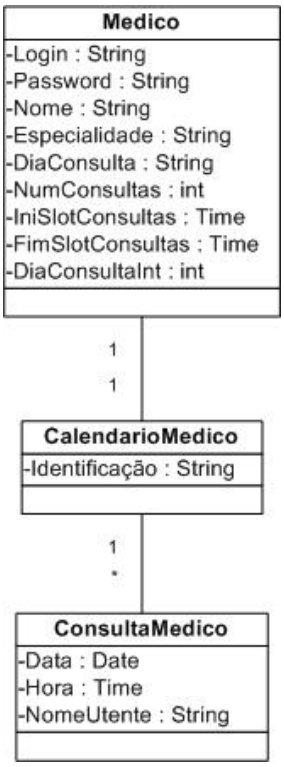


Fig 16: Diagrama de classes - Médico.

4.3.4. Base de Dados

Com a ajuda dos diagramas de classes construídos no ponto anterior, torna-se possível a construção dos diagramas de dados, ou mais formalmente, os diagramas de mapeamento relacional. Estes diagramas são a representação gráfica através de UML do modelo de dados usado para criar as bases de dados, incluindo as definições de chaves primárias e estrangeiras e as restrições de integridade de acordo com o modelo relacional.

De acordo com o diagrama de classes respectivo ao módulo dos utentes, é necessário atribuir três chaves primárias, uma por cada tabela, e duas chaves estrangeiras, uma por cada relação. Para a tabela Utente, define-se um novo atributo, IDUtente, que vai ser o identificador primário da tabela. Os atributos Login, Nome, Mec e Contacto têm a restrição *unique*, ou seja, identificam colunas que não podem possuir dados repetidos. Na tabela CalendarioUtente foi adicionado um atributo RefIdUtente, este atributo é em simultâneo a chave primária da tabela e a chave estrangeira que define a relação com a tabela Utente visto ser uma relação do tipo 1..1. Os mesmos conceitos foram aplicados à tabela ConsultaUtente, criando-se uma chave primária com o atributo IDConsulta, e uma chave estrangeira RefIdCalendarioUt que define a relação 1..* com a tabela CalendarioUtente.

Todos os atributos, independentemente da tabela em questão, são de carácter obrigatório, não sendo opcional o seu preenchimento aquando da adição de uma nova entrada. Os atributos relativos a chaves primárias e estrangeiras são gerados automaticamente.

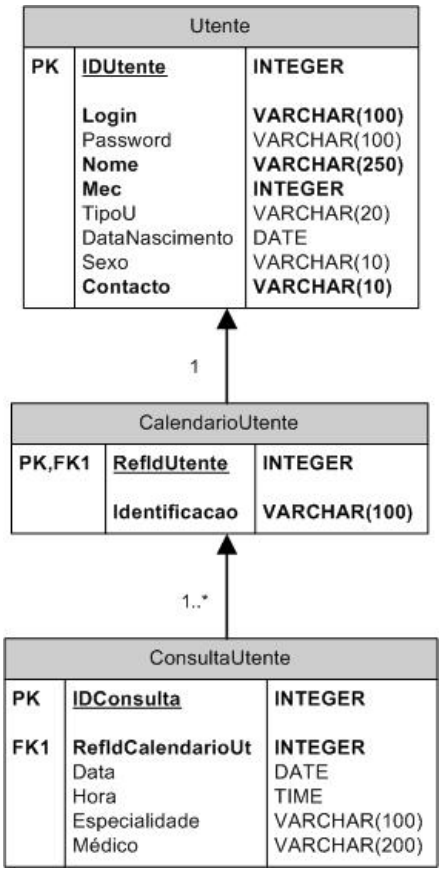


Fig 17: Diagrama de dados - Utente.

O diagrama de classes para o módulo dos funcionários dos SASUA é constituído por apenas uma classe, assim, obtém-se apenas uma tabela sem necessidade de criação de chaves estrangeiras. O atributo IDFuncSASUA é a chave primária definida para a tabela. Os atributos Login e Nome têm a restrição *unique*, sendo juntamente com o atributo Password obrigatórios ao adicionar uma nova entrada na tabela FuncSASUA.

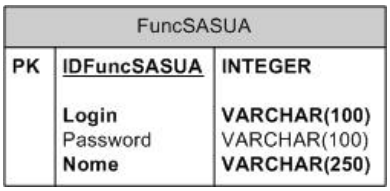


Fig 18: Diagrama de dados - Funcionários SASUA.

Relativamente ao diagrama de classes respectivo ao módulo dos médicos, a atribuição de chaves é semelhante ao definido para o módulo de utentes. Para a tabela Medico, define-se uma chave primária, IDMedico, e atribui-se a restrição *unique* aos atributos Login e Nome. Na tabela CalendarioMedico é adicionado um atributo RefIdMedico, sendo em simultâneo a chave primária da

tabela e a chave estrangeira que define a relação do tipo 1..1 com a tabela Medico. Para a tabela ConsultaMedico é definida uma chave primária RefIdConsulta, e uma chave estrangeira RefIdCalendarioMed que define a relação do tipo 1..* com a tabela CalendarioMedico.

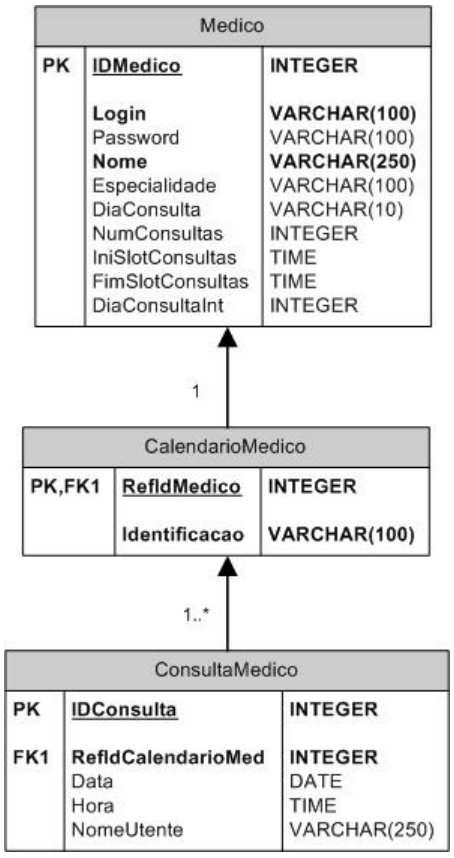


Fig 19: Diagrama de dados - Médico.

4.4. Interfaces

Para modelar o *design* das interfaces de utilizador foram usados ficheiros de estilo em formato CSS. Este formato permite definir tamanhos, cores, tabelas, cabeçalhos e todo um conjunto de aspetos visuais importantes numa aplicação web.

Como se trata de uma plataforma de gestão de dados em que a funcionalidade é de longe mais importante que o *design*, foram escolhidos tons neutros e uma palete de cores reduzida. O objetivo foi obter interfaces com aspeto funcional e sóbrio, evitando demasiadas misturas visuais e cores possivelmente chocantes à vista. Para ajudar na sua diferenciação, foi escolhido um tom diferente para cada módulo.

A nível de *design*, as interfaces têm como base um modelo simples e minimalista, um cabeçalho e rodapé, e um *container* com o conteúdo principal dividido em duas colunas. O menu

onde o utilizador pode aceder às funcionalidades localiza-se entre o cabeçalho e o conteúdo, tem uma direção horizontal e os submenus surgem sob a forma de *dropdowns* quando um *link* do menu principal é ativado com o clique do rato.

Fig 20: Página inicial do módulo de utentes.

4.5. Módulos

4.5.1. Etapas de desenvolvimento

Após o sistema ter sido modelado, e todos os seus elementos (internos e externos) terem sido identificados, é importante definir um plano de desenvolvimento por etapas. Delinear um projeto definindo objetivos ao longo do tempo ajuda a manter o trabalho organizado e a cumprir mais facilmente esses mesmos objetivos.

Apesar de a plataforma ser dividida em três módulos distintos e todos terem sido trabalhados individualmente, muitos dos passos e tarefas realizadas no desenvolvimento de um módulo repetem-se no desenvolvimento dos restantes, sendo ainda possível aproveitar trabalho de um módulo para

outro. O módulo de utentes foi o primeiro a ser desenvolvido, seguindo-se o de médicos e o de funcionários, no final foram trabalhados os três em simultâneo de modo a acertar pequenos detalhes e evitar discordâncias entre módulos.

O trabalho realizado na construção de cada módulo também foi organizado em pequenos passos. Cada funcionalidade de um módulo foi implementada numa página web específica, tornando assim mais fácil e intuitivo a navegação na plataforma quando um utilizador tem necessidade de realizar uma determinada ação. Para testar o funcionamento de cada uma das funcionalidades, foi por vezes necessário escrever alguns dados de forma manual nas bases de dados. A ordem de trabalhos dentro de um módulo seguiu então os seguintes passos:

- construção da base de dados e respectivas tabelas – JDBC;
- criação das classes e unidades de persistência associadas à base de dados – JPA;
- construção das páginas de estilos – ficheiros CSS;
- desenvolvimento da página e métodos associados ao login – XHTML e Java;
- testes;
- desenvolvimento da página e métodos associados a cada funcionalidade, incluindo possíveis regras de navegação entre páginas – XHTML e Java;
- testes (individuais para cada funcionalidade);
- implementação de verificações e mensagens de erro – XHTML e Java;
- documentação com javadoc em todos os ficheiros de código e respectivos métodos;
- testes finais.

4.5.2. Acesso e integridade dos dados

O bom funcionamento da plataforma tem como base a execução de algumas tarefas, nomeadamente no que diz respeito a verificações de dados e permissões de acesso. Estas tarefas são comuns aos três módulos.

Qualquer funcionalidade (independentemente do módulo em questão) em que é necessária a introdução de dados por parte do utilizador, está sujeita a uma verificação dos dados introduzidos antes da ação ser completada. Ao ser verificado *a priori* a validade e integridade da informação, reduz-se a probabilidade de corromper as bases de dados com dados inválidos evitando-se assim futuros problemas na aplicação. É importante salientar que no caso de falha destas verificações o utilizador é devidamente informado do erro cometido, não lhe sendo permitido prosseguir ou finalizar a ação pretendida até que o erro se encontre corrigido.

Algumas ações implicam o acesso às bases de dados, não só para leitura mas também para escrita. Como é crucial que os dados se encontrem atualizados em tempo real, a escrita ou edição de informação pode ter de ser realizada em simultâneo em tabelas associadas a dois módulos. Quando um determinado utilizador efetua, por exemplo, a desmarcação de uma consulta, os dados associados à mesma têm de ser removidos da tabela de consultas dos utentes e também da tabela de consultas dos médicos. Assim acontece com qualquer outra funcionalidade que implique a alteração de dados.

No que diz respeito aos dados das consultas, estes encontram-se apenas em tabelas de dados associadas aos utentes e aos médicos. Para evitar a sincronização de demasiados dados e manter a

integridade dos mesmos, os utilizadores do módulo direcionado para os funcionários do SASUA acedem aos dados de consultas gravados na tabela dos médicos. Esta opção foi tomada tendo também em consideração que os funcionários dos SASUA apenas podem aceder ao calendário de consultas de forma parcial, ou seja, estes apenas podem ver quais os *slots* temporários vagos ou ocupados não tendo qualquer acesso aos dados do utente que ocupa um determinado *slot.*

4.5.3. Módulo Utentes

O módulo de utentes destina-se a todos os utilizadores dos serviços do Núcleo de Saúde dos SASUA: alunos e funcionários docentes e não docentes. A funcionalidade mais importante que este módulo oferece aos seus utilizadores é a marcação de consultas *online*, contornando assim alguns problemas que surgem com o funcionamento atual do serviço.

O registo do utente na plataforma, ainda que não seja uma funcionalidade no uso diário do serviço, é obviamente importante e carece de explicação. O registo na plataforma é efetuado pelo utente aquando do primeiro acesso, sendo todos os seus dados importados da base de dados da Universidade (simulada pela base de dados que contém os atributos mencionados na Fig 13). Esta base de dados não só serve para retornar os dados do utilizador, mas também para verificar a veracidade do utilizador como membro da universidade.

Após o utilizador ter sido devidamente identificado, é-lhe pedido que confirme a validade dos seus dados antes de confirmar o registo, sendo este passo realizado uma única vez. Apenas utilizadores registados podem aceder às funcionalidades do módulo. Os passos efetuados para o login e registo do utilizador (no 1º acesso à plataforma) são demonstrados no seguinte diagrama de atividades (Fig 21).

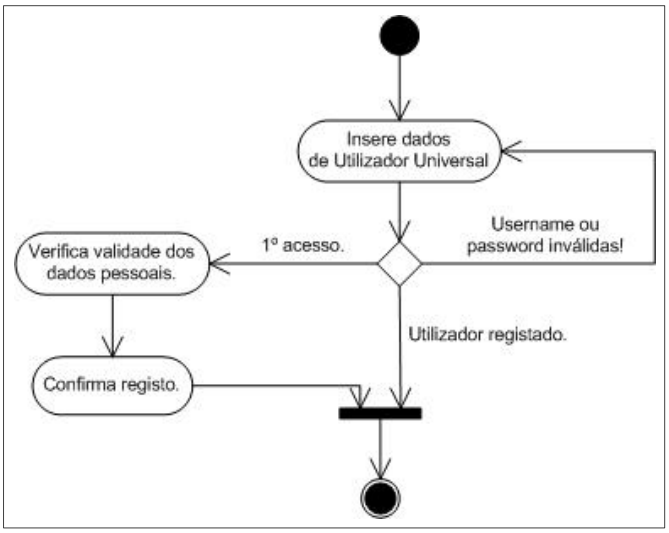


Fig 21: Login e registo - Utente.

O utente apenas tem permissão para modificar o seu contacto. Esta restrição evita que o utilizador tente alterar dados que possam corromper a integridade do sistema, como por exemplo, nome, número mecanográfico ou login. Quando o utente efetua esta alteração, o conteúdo introduzido é verificado, permitindo apenas cadeias de caracteres que formem um número de contacto válido, fixo ou móvel. Não são aceites letras ou outros caracteres não numéricos. Se a cadeia de caracteres introduzida for válida, o sistema informa o utilizador com uma mensagem de sucesso, e a tarefa de edição de dados é dada como terminada (Fig 22).

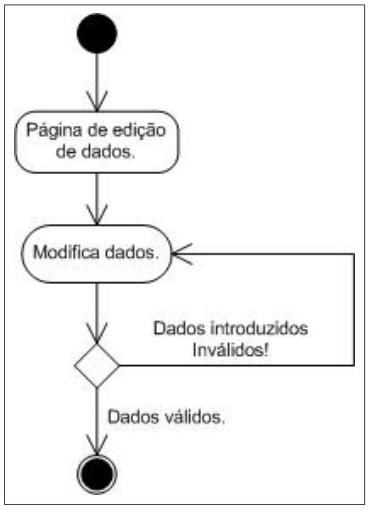


Fig 22: Edição de dados pessoais - Utente.

Quando o utente pretende marcar uma consulta, é-lhe disponibilizado um formulário para o preenchimento do nome do médico e especialidade pretendida, juntamente com a data de consulta. O formulário é composto por elementos *dropdown* com dados pré-definidos, evitando assim que o utente tenha de preencher os dados todos manualmente. Após ter sido selecionado o médico e respectiva especialidade, o ano e o mês, o sistema mostra ao utente os dias desse mês em que ainda existem vagas para consultas (de acordo com o dia da semana estabelecido para a especialidade). O utente seleciona assim o dia que mais lhe convém e confirma a marcação da consulta, sendo a mesma registada automaticamente no calendário do médico em questão (Fig 23).

Para auxiliar esta tarefa, acima do formulário encontra-se a tabela de consultas médicas com todos os médicos existentes, respectivas especialidades, dias de consulta e horário das mesmas.

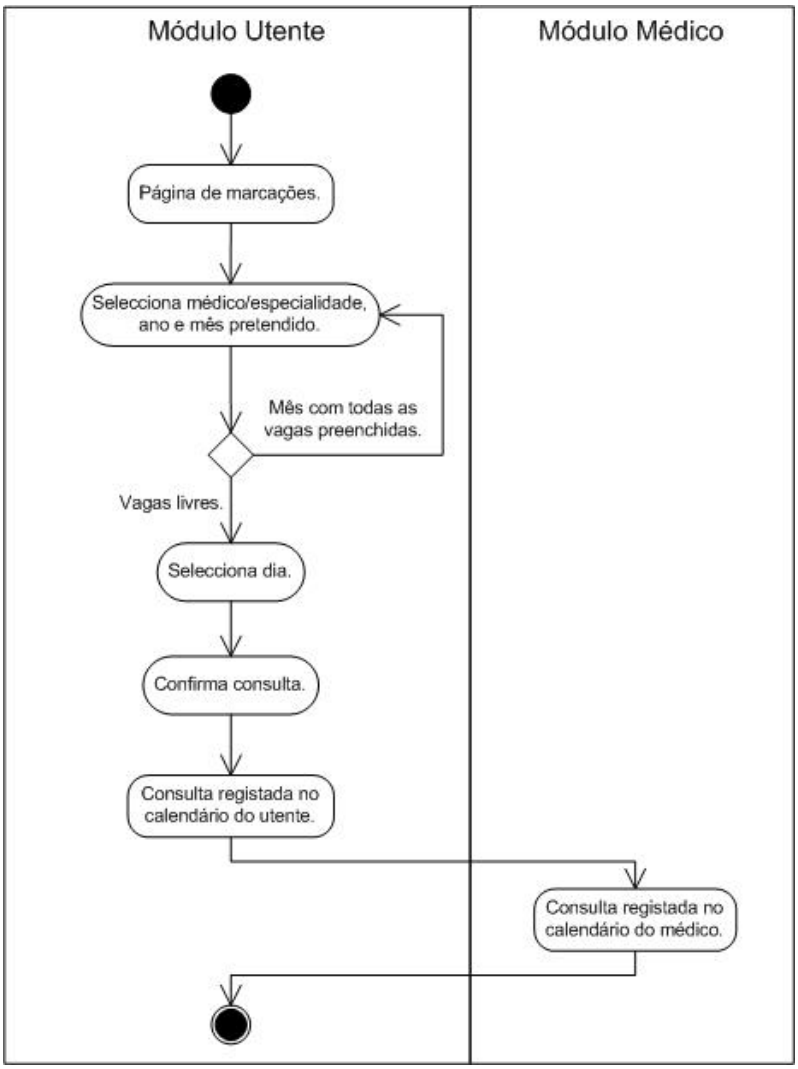


Fig 23: Marcação de consultas - Utente.

Para desmarcar uma consulta, o utente tem disponível uma página com o seu calendário de consultas completo. O utente apenas tem de selecionar a consulta (ou consultas) que pretende desmarcar, marcando para isso a *checkbox* correspondente à consulta em questão. Ao desmarcar a consulta, o sistema considera automaticamente que a vaga foi libertada para outro possível utente.

Após a desmarcação, o utente é redirecionado para a página de calendário já atualizada (Fig 24). Nesta página é mostrada uma tabela com o médico e especialidade de cada consulta, bem como o dia em que a consulta se encontra marcada. A tabela mostra apenas consultas do dia atual ou marcadas para datas posteriores.

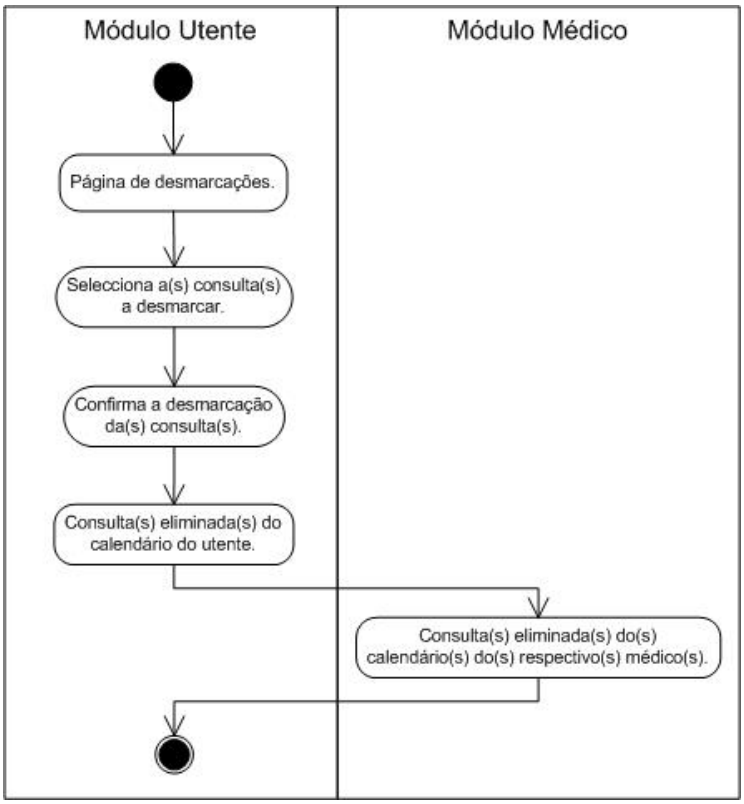


Fig 24: Desmarcação de consultas - Utente.

Para sincronizar o calendário de consultas da PGCO com um calendário Google, o utente tem ao seu dispor uma página específica para essa funcionalidade. É pedido ao utente que efetue login na sua conta Google, e que introduza os mesmos dados de acesso num formulário próprio para o efeito, presente na página. Após o preenchimento do formulário e dos dados introduzidos terem sido verificados como válidos, um calendário Google embebido na página mostra as consultas marcadas para o utente (Fig 26). Sempre que são marcadas novas consultas, o utente deve fazer uma nova sincronização, sendo o calendário Google atualizado com as consultas em falta.

Na primeira sincronização de calendários, caso nenhum calendário exista, o sistema cria um novo na conta Google do utente. Cada consulta criada nesse calendário é identificada como "SASUA - <nome da especialidade> com <nome do médico>, às <hora da consulta>".

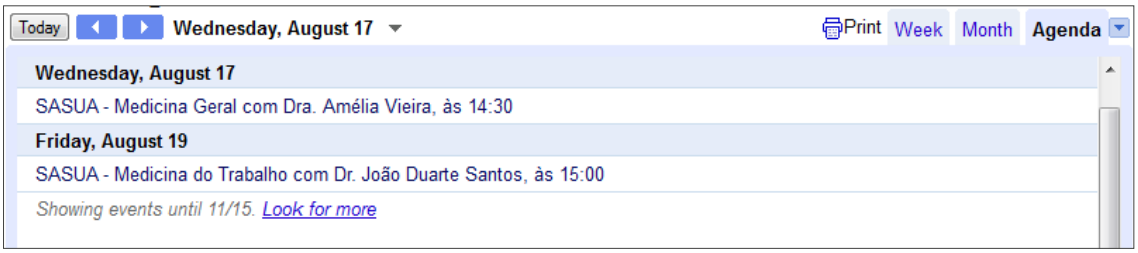


Fig 25: Exemplo de calendário Google embebido com consultas (formato Agenda).

Fig 26: Sincronização com calendário Google - Utente.

4.5.4. Módulo Funcionários SASUA

O módulo de funcionários é direcionado aos funcionários dos SASUA (secretaria e funcionária do posto médico). Este é o módulo com menos permissões a nível de visualização do calendário médico, os funcionários não têm acesso aos dados dos utentes com consultas marcadas. Também não lhes é permitida a marcação de consultas, sendo essa tarefa destinada apenas a médicos e utentes.

Os utilizadores deste módulo não possuem nenhuma forma de registo no mesmo. Para evitar que utilizadores não autorizados tenham acesso às funcionalidades, é da responsabilidade dos Serviços de Acão Social adicionar à base de dados apenas os devidos funcionários.

Uma das funcionalidades mais importantes para um funcionário dos SASUA é o registo de novos médicos, e a consequente definição do seu horário de consultas no sistema. Para esta funcionalidade, o funcionário tem ao seu dispor uma página com um formulário onde deve definir o nome do médico (antecedido de Dr. ou Dra., conforme o género), o nome da especialidade, o número máximo de consultas a exercer (por dia), o dia da semana em que serão exercidas as consultas, e as horas de início e fim para as mesmas. Para estes últimos três campos, o utilizador tem à disponibilidade vários menus *dropdown* com valores pré definidos.

Após o preenchimento de todos os campos necessários, o funcionário pode então confirmar a adição de um novo médico no sistema. Se a ação for bem-sucedida, o funcionário é redirecionado para uma página que lista todos os médicos existentes no sistema, e os dados relativos ao horário de consultas de cada um.

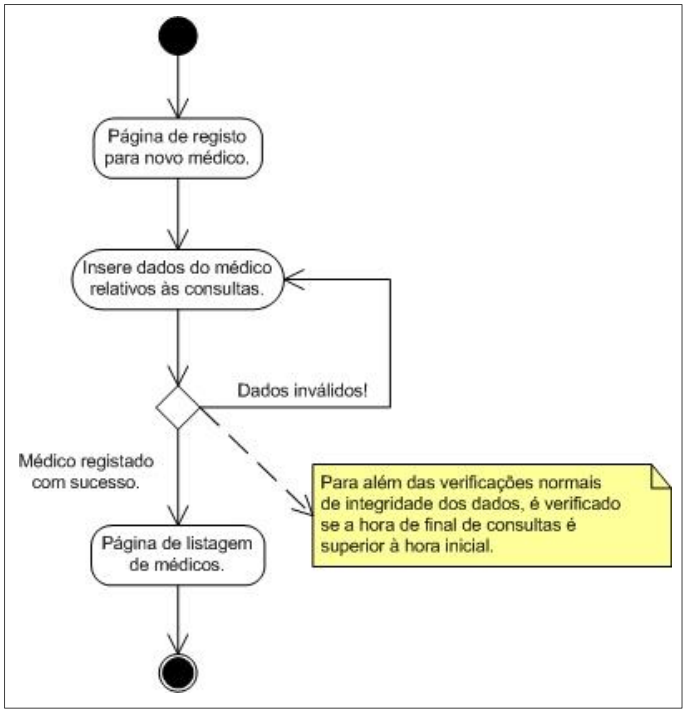


Fig 27: Registo de um novo médico - Funcionário SASUA.

Sempre que necessário, o funcionário dos SASUA pode editar os dados do registo de um médico. Os dados passíveis de alteração são o dia da semana estabelecido para as consultas, a hora de início e a hora de fim. Para esta tarefa, o funcionário tem ao seu dispor uma página que contém uma tabela com o horário de consultas de todos os médicos no sistema, e abaixo da tabela um formulário que lhe permite alterar os dados mencionados anteriormente. Esta funcionalidade pode ser modelada pelo diagrama já apresentado para a edição dos dados do utente, referenciado pela figura Fig 22.

O funcionário pode também remover um médico do sistema. Nesse caso é-lhe disponibilizada uma tabela com a listagem de médicos registados, e uma *checkbox* ao lado de cada nome, sendo apenas necessário selecionar a *checkbox* específica ao(s) médico(s) a remover.

No final de ambas as tarefas, o funcionário dos SASUA é redirecionado para a página atualizada de listagem de médicos e respectivos horários de consultas, tal como sucedia com a funcionalidade relativa à adição de médicos no sistema.

A desmarcação de consultas permitida aos funcionários dos SASUA, foi pensada sobretudo como uma funcionalidade de apoio aos médicos, sendo apenas possível a desmarcação do conjunto

de consultas marcadas para um determinado dia. O objetivo é o de permitir a desmarcação de consultas de um médico no caso de este não poder comparecer ou não poder aceder à PGCO. Para a realização desta tarefa, o funcionário seleciona na página apropriada, o nome do médico em questão, e a data das consultas a desmarcar. Ao confirmar a desmarcação das consultas, o funcionário é redirecionado para a página de envio de notificações. A notificação é escrita pelo funcionário e enviada por e-mail a todos os utentes cuja consulta foi desmarcada, sendo o contacto de e-mail de cada utente (igual ao login de Utilizador Universal) adicionado automaticamente pelo sistema, não sendo visíveis estes contactos na página de notificação.

As funcionalidades de desmarcação de consultas e envio de notificações foram modeladas no mesmo diagrama de atividades, demonstrado pela figura Fig 28.

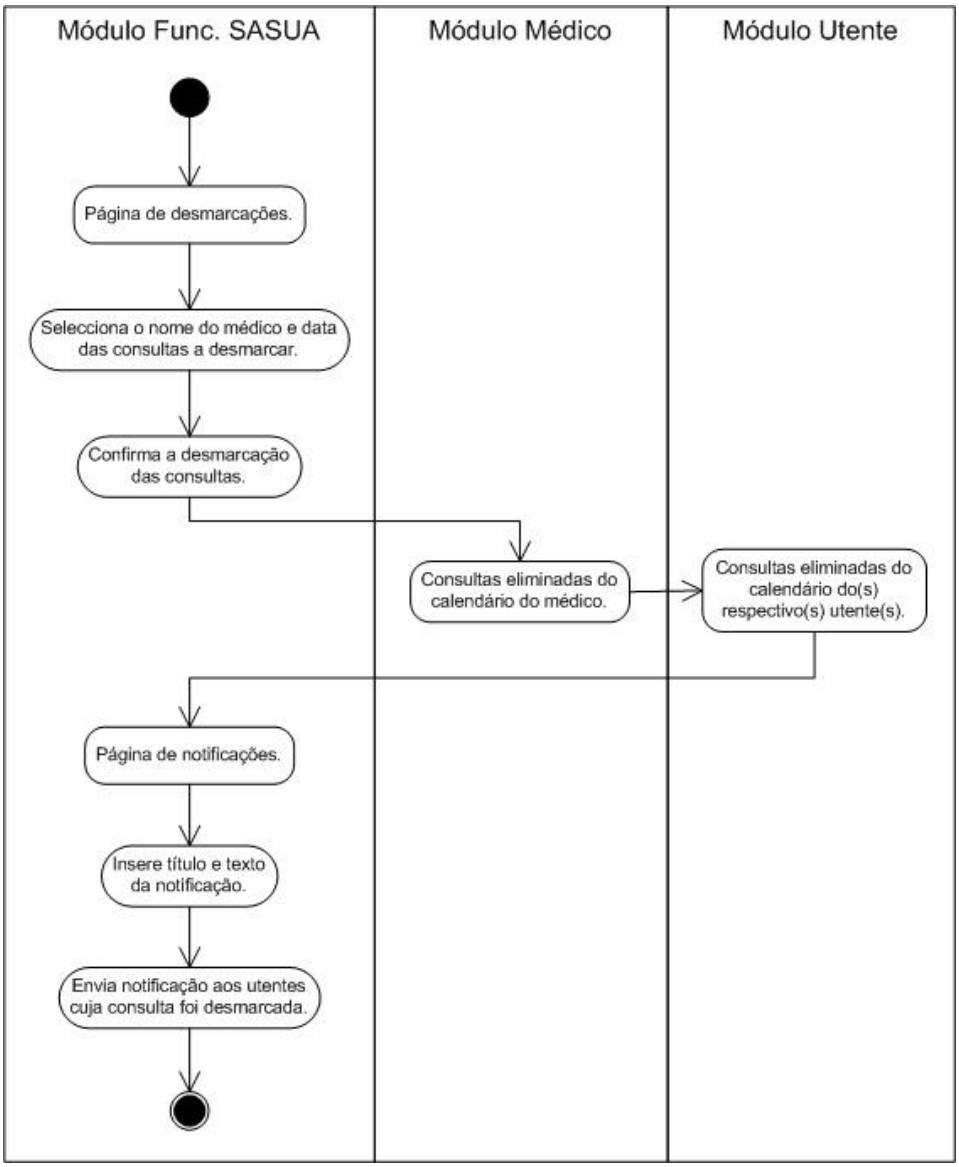


Fig 28: Desmarcação de consultas e envio de notificações - Funcionário SASUA.

4.5.5. Módulo Médicos

Este módulo destina-se aos médicos que exercem as consultas gratuitas. No conjunto das funcionalidades oferecidas, é proporcionada uma grande vantagem. Estes utilizadores passam a ter acesso direto à sua agenda de consultas a qualquer momento, permitindo-lhes assim poder preparar as consultas previamente (útil em especialidades que requerem um seguimento periódico do utente).

Ao contrário do que acontecia com os utentes, os médicos não se registam na plataforma, sendo essa tarefa da responsabilidade dos funcionários dos SASUA (como referido no ponto anterior). Os médicos têm acesso direto ao seu módulo a partir do momento que os seus dados são adicionados à base de dados médica.

O médico está autorizado a marcar consultas para os seus utentes sempre que necessário. Para realizar essa tarefa o médico deve preencher um formulário onde indica a data da consulta (ano, mês e dia) e o nome completo do utente a quem a mesma se destina. À semelhança dos formulários já referidos anteriormente, o médico seleciona a data da consulta através de menus *dropdown* com valores pré-definidos, sendo mostrado após a seleção do ano e mês, os dias desse mês que ainda possuem vagas para marcação. Após o preenchimento, e após ter sido validado o nome do utente (nome completo correto, sendo necessário que o utente esteja registado no sistema), o médico pode então confirmar a marcação da consulta (Fig 29). A consulta fica a partir desse momento marcada não só no calendário do médico, mas também no calendário do utente em questão.

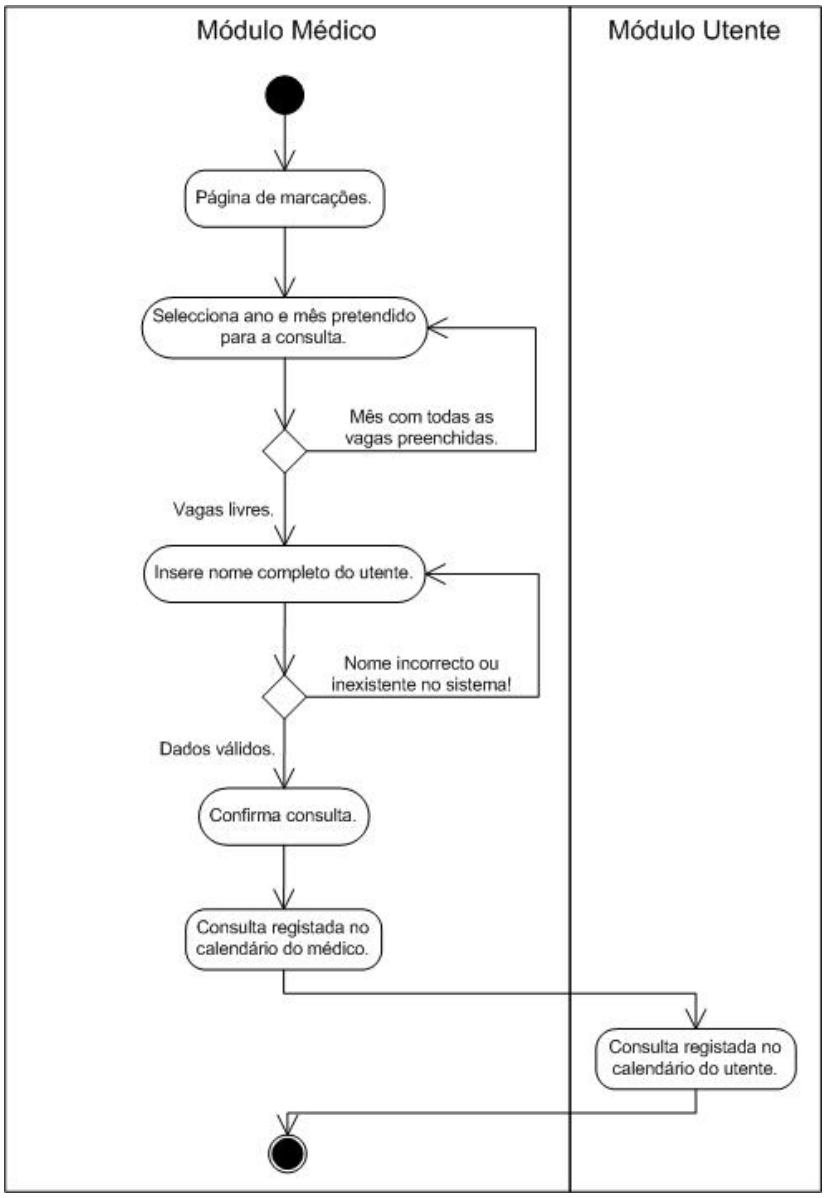


Fig 29: Marcação de consultas - Médico.

A desmarcação de consultas procede-se de forma semelhante ao que acontecia para os funcionários dos SASUA. Considerando que o utente pode desmarcar as suas próprias consultas, optou-se por apenas permitir aos médicos a desmarcação de consultas em bloco, ou seja, a desmarcação de todas as consultas para um determinado dia. O médico seleciona o ano e mês das consultas que pretende desmarcar, sendo-lhe de seguida mostrado o calendário de marcações para os dias desse mesmo ano e mês. Após selecionar o dia pretendido, o médico pode então confirmar a desmarcação das consultas. Estas são removidas automaticamente do calendário do médico, e também do calendário dos utentes afetados.

Também em semelhança com o que se sucedia para os funcionários dos SASUA, ao confirmar a desmarcação das consultas, o médico é redirecionado para a página de envio de notificações. A notificação é escrita pelo médico e enviada por e-mail a todos os utentes cuja consulta foi desmarcada, sendo o contacto de e-mail de cada utente (igual ao login de Utilizador Universal) adicionado automaticamente pelo sistema, não sendo visíveis estes contactos na página de notificação (Fig 30).

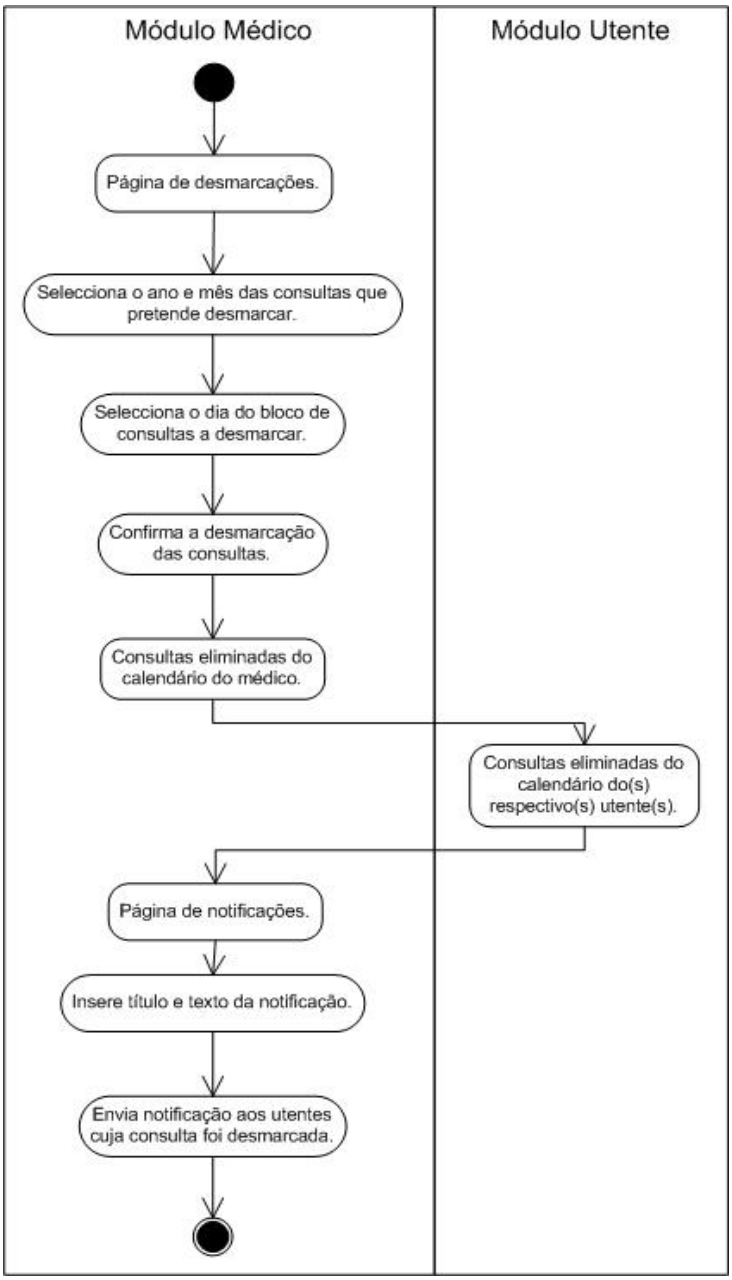


Fig 30: Desmarcação de consultas - Médico.

A visualização do calendário permite ao médico ver todos os detalhes das consultas, incluindo os dados dos utentes com consultas marcadas. Quando o médico acede ao seu calendário de consultas, e seleciona a data das consultas que pretende ver, o sistema disponibiliza o número de consultas marcadas para cada dia do mês e ano selecionados. De seguida, se assim o desejar, o médico pode ver os detalhes das consultas clicando num botão específico para esse efeito. Nesse momento o médico é redirecionado para uma página onde se encontram os dados de todos os utentes com consulta marcada para o dia selecionado (Fig 31).

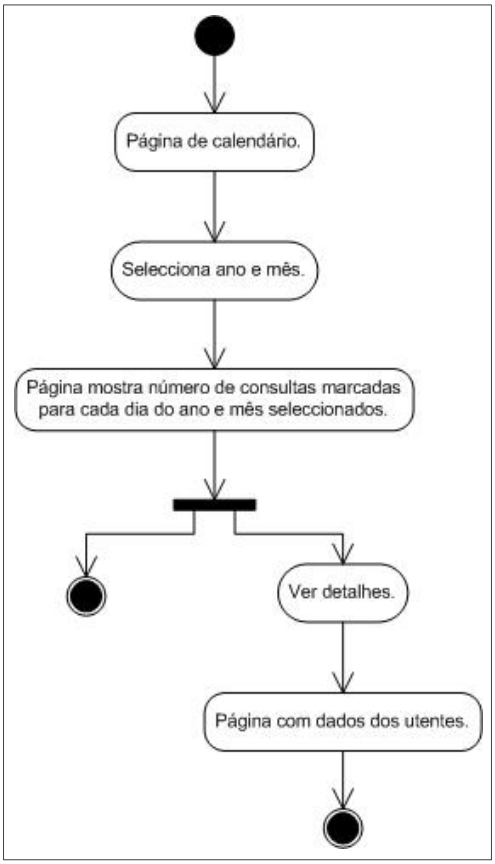


Fig 31: Visualização de calendário e dados dos utentes - Médico.

4.6. Interfaces de Utilizador

Após toda a modelação do sistema ter sido especificada, e todas as funcionalidades de cada módulo terem sido explicadas detalhadamente, seguem-se vários *print-screens* das interfaces de utilizador que permitem demonstrar o uso das diversas funcionalidades.

A página de registo mostra ao utilizador todos os dados retornados da base de dados de Utilizador Universal. Nesta página, o utilizador deve confirmar o seu registo na plataforma caso todos

os seus dados se verifiquem corretos, ou anular a operação em caso contrário. Após o registo o utilizador pode então efetuar *login* na plataforma e começar a usar as suas funcionalidades (Fig 32).

Fig 32: Página de registo de um utente.

Quando o utente pretende marcar uma nova consulta, é-lhe apresentada uma página com uma tabela informativa com todos os médicos disponíveis e respetivos horários de consulta, e de seguida um formulário de preenchimento rápido, como demonstrado na figura Fig 33. Nesse formulário, o utente deve definir o nome do médico (já associado à respetiva especialidade), o ano e o mês da consulta. Com estes dados o utente pode verificar a disponibilidade de vagas para os dias do ano e mês pretendidos, e caso algum dos dias se adeque à sua disponibilidade, o utente pode então efetivar a marcação da consulta.

Se o sistema encontrar alguma irregularidade no preenchimento do formulário, ou não conseguir efetivar a marcação da consulta, informa o utente do sucedido com uma mensagem de erro no topo do conteúdo central da página. Caso a consulta fique corretamente registada, o utente recebe uma mensagem de sucesso.

Fig 33: Marcação de consulta para um utente.

Quando o utente pretende visualizar o seu calendário de marcações, a plataforma disponibiliza-lhe uma página específica para o mesmo. Nesta página é mostrada a listagem de consultas com data igual ou superior à atual, ordenadas por data, e com a informação do nome do médico, especialidade e hora de consulta (Fig 34).

Fig 34: Calendário de marcações de um utente.

Quando um funcionário dos SASUA pretende registar um novo médico no sistema, deve preencher o devido formulário com o nome do médico, a especialidade, o número máximo de

consultas a exercer em cada dia, o dia semanal das consultas e as horas de início e fim das mesmas. Se o sistema não detetar nenhuma falha ou incorreção no preenchimento dos dados necessários, o funcionário pode confirmar o registo do novo médico, caso contrário é-lhe mostrada uma mensagem de erro assinalando a falha cometida (Fig 35).

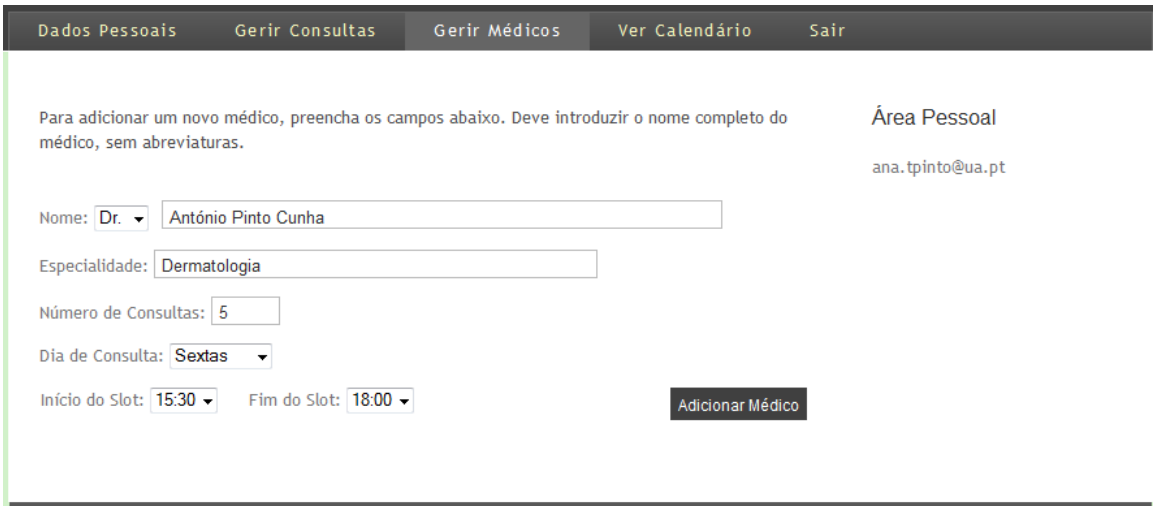


Fig 35: Registo de um novo médico no sistema.

Como referido anteriormente na especificação das funcionalidades, os funcionários dos SASUA não têm acesso a todas as informações do calendário de marcações dos médicos. Assim, sempre que um funcionário tiver necessidade de aceder ao calendário de um determinado médico, deve definir no formulário que lhe é apresentado o nome do mesmo, o ano e mês. Como demonstrado na figura seguinte (Fig 36), com esta ação o funcionário pode então ver o número de *slots* ocupados para os dias do ano e mês que definiu.

Fig 36: Calendário de marcações para o funcionário dos SASUA.

Quando um funcionário dos SASUA efetua a desmarcação de consultas é redirecionado para a página de envio de notificações. O sistema reconhece automaticamente o e-mail dos utentes a quem a consulta foi desmarcado, tendo o funcionário de preencher apenas os campos respetivos ao nome do médico e especialidade, a data da consulta desmarcada e o texto de notificação. A página para o envio de notificações demonstrada na figura Fig 37 é igual tanto para funcionários dos SASUA como para médicos.

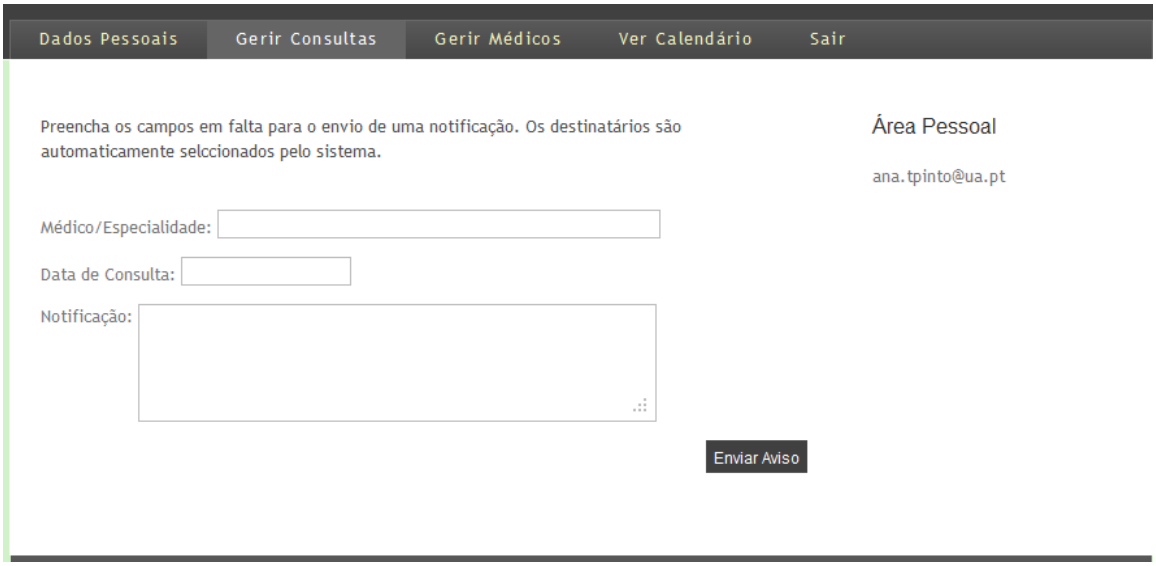


Fig 37: Página de envio de notificações (funcionários SASUA e médicos).

As figuras seguintes (Fig 38 e Fig 39) apresentam o formato de visualização de calendário para um médico e a página de visualização de dados do utente, respetivamente. As ações efetuadas para a visualização do calendário são semelhantes ao que acontecia para o funcionário dos SASUA. Para aceder aos dados dos utentes com consulta marcada para um determinado dia, o médico apenas tem de selecionar o dia pretendido e clicar em "Ver Detalhes".

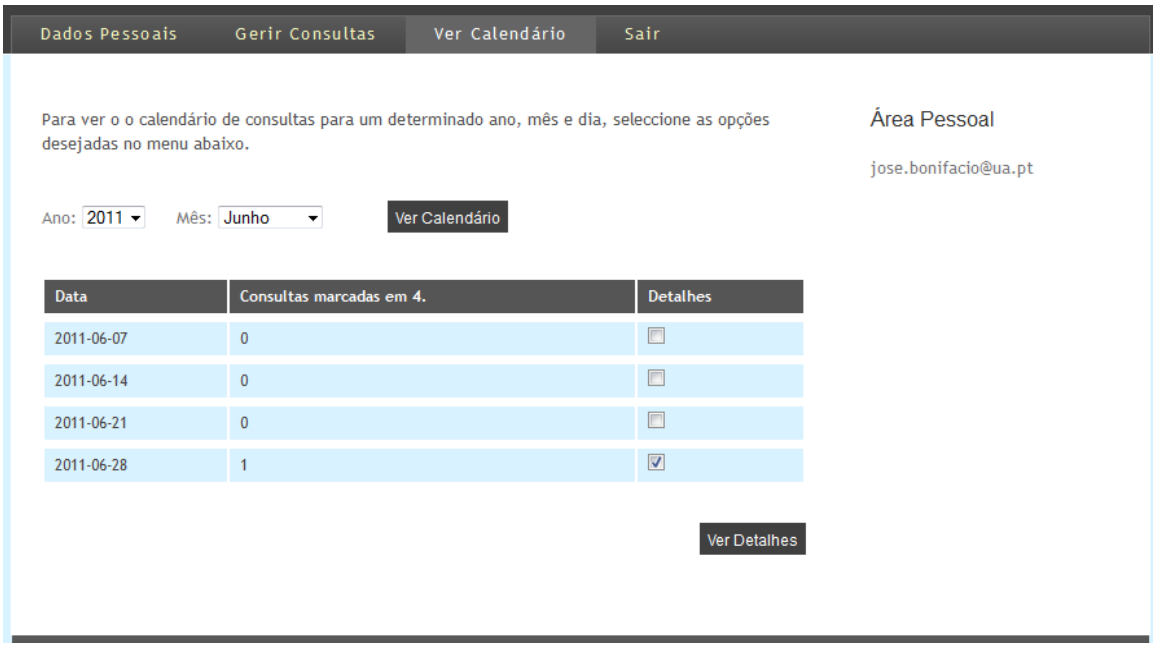


Fig 38: Calendário de marcações para o médico.

Fig 39: Detalhes do calendário do médico - dados do utente.

5. Conclusões e Trabalho Futuro

5.1. Conclusões

A oportunidade de realizar um projeto de aplicação prática e com o objetivo de satisfazer as necessidades de um grupo abrangente de utilizadores, foi o necessário para ganhar motivação à realização desta dissertação.

Conhecer o funcionamento geral do serviço de consultas gratuitas oferecidas pelos SASUA, bem como os métodos atuais de gestão do mesmo, permitiu identificar um possível caso de estudo e o consequente projeto aplicacional. O objetivo era o de analisar o caso de estudo em mãos, perceber quais as necessidades dos vários utilizadores do serviço, e criar uma plataforma de gestão que satisfizesse essas mesmas necessidades. Após todo o trabalho desenvolvido e os resultados conseguidos, é agora possível fazer algumas considerações finais que concluem a realização do projeto.

A análise de requisitos inicial elaborada com o intuito de conhecer mais a fundo todos os problemas existentes, permitiu identificar em específico quais os futuros utilizadores da plataforma, e identificar um conjunto de funcionalidades a implementar para os mesmos. Toda esta análise foi crucial para entender o problema existente e para começar a pensar em soluções a adotar.

As escolhas feitas ao nível das tecnologias de suporte necessárias à elaboração das soluções pensadas permitiram focar as atenções nos aspetos funcionais da plataforma, conseguindo-se obter os resultados desejados.

Ao longo das diversas etapas de desenvolvimento, foi necessário e importante ter em conta o objetivo principal da plataforma e o perfil dos seus futuros utilizadores. O desenvolvimento foi realizado seguindo alguns princípios de simplicidade, funcionalidade e usabilidade, princípios definidos pelas heurísticas de usabilidade de Jakob Nielsen [**36**], assim, foi dada maior importância às funcionalidades em si e à sua realização de forma simples e prática por parte dos utilizadores, e dada uma menor importância ao aspeto e *design* das interfaces. Conseguiu-se com estas linhas de orientação, uma plataforma de simples utilização com um *design* funcional, minimalista e sóbrio.

O desenvolvimento do projeto nem sempre correu como esperado, e ao longo das várias etapas foi necessário ultrapassar diferentes obstáculos. As versões mais recentes de Glassfish e Netbeans causavam conflitos no acesso às bases de dados em *runtime*, tendo sido necessário reverter o primeiro para a versão estável anterior. À medida que as funcionalidades foram sendo desenvolvidas, foi necessário retificar pormenores ao nível das tabelas de dados (mais atributos ou alteração do tipo de dados dos mesmos), o que convergiu em alterações em várias partes do código. Também à medida que as funcionalidades implementadas iam sendo testadas, foi sempre necessário corrigir alguns erros e melhorar alguns aspetos considerados importantes. Todos estes obstáculos encontrados contribuíram para aumentar a duração do processo de desenvolvimento, mas ao mesmo

tempo, e também importante, para uma maior aprendizagem e maturidade no desenvolvimento de um projeto e de todos os aspetos inerentes.

As funcionalidades necessárias identificadas de início foram totalmente implementadas, e permitem agora um novo leque de vantagens aos futuros utilizadores. As falhas sentidas pelos utentes do serviço de consultas podem agora ser solucionadas, e os restantes utilizadores (médicos e funcionários) podem agora também fazer uma melhor gestão dos dados e aumentar a sua produtividade. O facto de o sistema estar preparado para a integração com as bases de dados de alunos e funcionários da universidade é uma mais-valia, permitindo a possível integração futura com outros serviços e plataformas da universidade.

5.2. Trabalho Futuro

5.2.1. Testes de Usabilidade, Esforço e Funcionalidades

Para garantir a total funcionalidade e integridade da plataforma PGCO, é necessário que esta seja submetida a vários testes por parte de um grupo de utilizadores. Com um conjunto pré-definido de testes a realizar através de guiões de usabilidade e funcionalidade, será possível verificar se todos os módulos se comportam como esperado, se não existem falhas no acesso às bases de dados, se as interfaces são usáveis e apelativas ao utilizador, e se com vários acessos em simultâneo o sistema se mantém íntegro e estável.

5.2.2. Integração de Sistemas

Como já referido como uma das mais-valias do sistema, este encontra-se preparado para a integração com as bases de dados da universidade. Apesar do sistema estar de momento integrado com uma base de dados simulada, para que o mesmo permita uma total funcionalidade de acesso é necessário que a integração seja efetivamente realizada com as base de dados devidas através de outros sistemas disponíveis na universidade.

5.2.3. Funções de Backoffice

Como referido anteriormente (secção 4.5.4), os funcionários dos SASUA com acesso ao sistema são adicionados diretamente na base de dados. Como trabalho futuro, poderiam criar-se as funcionalidades administrativas necessárias à gestão de funcionários com acesso ao sistema, desenvolvendo-se para isso um pequeno serviço de *backoffice* ao módulo de funcionários.

Também relevante seria a construção de uma página que disponibilizasse as várias estatísticas de utilização da plataforma pelos diversos utilizadores.

5.2.4. Gestão de Dados Clínicos

Com o alargamento do Centro de Saúde Universitário, foi pensado em desenvolver-se uma plataforma de apoio às consultas médicas. A ideia da construção desta plataforma surgiu no mesmo contexto da PGCO, com o objetivo de fornecer um apoio digital, centralizado e mais eficiente à gestão das consultas, mas desta vez direcionado à realização das consultas em si, à sua gestão e à gestão dos dados clínicos dos utentes (informações, prescrição de medicamentos, exames).

Apesar de não ter existido tempo suficiente para o desenvolvimento das duas plataformas, foi realizado algum trabalho de pesquisa de modo a perceber melhor quais as aplicações já existentes na mesma área e qual o seu modo de funcionamento.

5.2.4.1. EMR e EHR

Quando se fala em informatização de dados médicos, é necessário primeiro ter conhecimento dos dois acrónimos mais usados nesse processo e da sua definição, pois ainda que os nomes sejam semelhantes, ambos definem conceitos bastante diferentes e que é importante perceber.

EMR, o que em português se pode traduzir como Registos Médicos Eletrónicos, é um conceito que engloba todo o ambiente aplicacional necessário à informatização dos dados. As organizações que prestam cuidados de saúde, os repositórios de dados, os sistemas de apoio a decisões médicas, os sistemas de controlo de vocabulário, a documentação clínica e farmacêutica, todos estes elementos fazem parte de um EMR. Todos os dados presentes num EMR são a prova ou o registo legal de todo o processo clínico de um paciente perante a organização que lhe prestou os cuidados de saúde.

EHR, ou traduzido, Registos de Saúde Eletrónicos, são os registos médicos de cada paciente. Incluem um sumário de cada visita a uma unidade de cuidados médicos, registo de doenças, internamentos, medicação e toda a informação necessária a possíveis cuidados futuros. São estes os registos que numa organização de cuidados de saúde unificada (como já existe nos E.U.A.), passam de unidade médica em unidade médica sempre que o paciente assim o necessitar. [**37**]

5.2.4.2. Vantagens e Desvantagens

As vantagens que advêm do uso de registos eletrónicos são bastante claras, em qualquer área, mas sobretudo na área da saúde onde o acesso rápido à informação é crucial. E é importante referir que esta tecnologia é vantajosa não só para o paciente e para as unidades de saúde, mas também para a população em geral, como é referido mais à frente.

Analisando tudo o que os EMRs e EHRs envolvem, e toda a informação que é manipulada no processo, estas são as vantagens que se podem verificar [**38**]:

- Disponibilidade de informação segura e fidedigna em tempo real e a qualquer instante;
- Acesso remoto à informação;

- Integração com sistemas externos de monitorização e exames médicos, mesmo externamente à unidade de saúde;
- Integração com unidades farmacêuticas;
- Introdução facilitada e fidedigna de dados através de controlo de vocabulário e documentação;
- Facilidade no acesso à informação do paciente (doenças, internamentos, medicação, hereditariedade);
- Disponibilidade de ferramentas para monitorização e trabalho em equipa de médicos e enfermeiros;
- Disponibilidade de informação para estatísticas, pesquisas, melhorias de qualidade e iniciativas para a saúde pública;
- Disponibilidade de informação para ensaios clínicos.

É importante ainda assim referir que estes sistemas também têm as suas desvantagens, maioritariamente relacionadas com custos [**39**]. Nem todas as unidades de prestação de serviços de saúde estão suficientemente equipadas para suportar esta tecnologia, isto acarreta um custo inicial bastante significativo na sua implementação, tanto a nível de hardware como software. A manutenção dos serviços ao longo do tempo é também uma despesa a ter em consideração.

A outra desvantagem a considerar, prende-se com questões de segurança dos dados. No caso de a informação ser armazenada em servidores remotos (não localizados dentro das imediações da unidade de saúde), esta pode ser acedida por entidades terceiras, os provedores de serviços.

5.2.4.3. Atualidade

Os E.U.A. são atualmente o país onde existe o maior número de unidades de saúde já equipadas com tecnologia para EMR, existindo mesmo uma legislação para a proteção dos dados dos utentes. Foram também estabelecidos vários *standards* para a comunicação de informação e troca de informação entre sistemas EMR diferentes.

Na Europa, esta tecnologia é usada amplamente no Reino Unido e na Estónia, países onde já existe um sistema de saúde unificado. Muitos outros países já possuem plataformas eletrónicas para gestão de dados de saúde (como é o exemplo da Rede Telemática para a Saúde, 2.4.1), mas não ao nível de um EMR, principalmente no que diz respeito a trocas de informação e exames médicos entre várias unidades de saúde [**40**].

6. Bibliografia

[1] "Regulamento Orgânico dos Serviços de Acção Social da Universidade de Aveiro", 2010. [Online] Disponível em: http://www2.sas.ua.pt/incoming/capas/RO%20SASUA.pdf.

[2] "Linha Universidade de Aveiro". [Online] Disponível em: http://www.ua.pt/sas/lua/, acedido a Outubro 2011.

[3] "Rede Telemática Saúde", 2007. [Online] Disponível em: http://www.rtsaude.pt/paginas_frontoffice/frontoffice_home.php, acedido a 11 Outubro 2011.

[4] uaonline, "Rede Telemática de Saúde entrou em funcionamento". [Online] Disponível em: http://uaonline.ua.pt/detail.asp?c=21627, acedido a 11 Outubro 2011.

[5] Ministério da Saúde, "Sistema de Informação Novo Programa Nacional de Promoção da Saúde Oral - Relatório Técnico", 2008.

[6] Ordem dos Médicos Dentistas, "PNPSO - Saúde Oral para Grávidas e Saúde Oral para Idosos",. [Online] Disponível em: http://www.omd.pt/chequedentista/chequedentista-infoclasse.pdf.

[7] "VitalJacket". [Online] Disponível em: www.vitaljacket.com.

[8] Sun Microsystems, 1996. [Online] Disponível em: http://java.sun.com/docs/white/langenv/Intro.doc2.html#334, acedido a 6 Agosto 2011.

[9] HAGGAR P, "Java Bytecode - Understanding bytecode makes you a better programmer", 2001. [Online] Disponível em: http://www.ibm.com/developerworks/ibm/library/it-haggar_bytecode/, acedido a 14 Outubro 2011.

[10] Oracle Corporation, 1995. [Online] Disponível em: http://download.oracle.com/javase/tutorial/getStarted/intro/definition.html, acedido a 6 Agosto 2011.

[11] SESTOFT P, "Numeric performance in C, C# and Java", IT University of Copenhagen, 2010. [Online] Disponível em: http://www.itu.dk/~sestoft/papers/numericperformance.pdf, acedido a 25 Agosto 2011.

[12] Cherrystone Software Labs, "Algorithmic Performance Comparison Between C, C++, Java and C# Programming Languages", 2010. [Online] Disponível em: http://www.cherrystonesoftware.com/doc/AlgorithmicPerformance.pdf, acedido a 25 Agosto 2011.

[13] Oracle Corporation, "javadoc - The Java API Documentation Generator", 2004. [Online] Disponível em: http://download.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html, acedido a 20 Agosto 2011.

[14] Sun Microsystems, "How to Write Doc Comments for the Javadoc Tool", 2004. [Online] Disponível em: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html, acedido a 20 Agosto 2011.

[15] Object Management Group, "UML Resource Page". [Online] Disponível em: http://www.uml.org/, acedido a 17 Outubro 2011.

[16] Object Management Group, "OMG Unified Modeling Language (OMG UML) Superstructure", 2009. [Online] Disponível em: http://www.omg.org/spec/UML/2.2/Superstructure/PDF/, acedido a 17 Outubro 2011.

[17] Oracle Corporation, 1997. [Online] Disponível em: http://download.oracle.com/javase/tutorial/jdbc/overview/index.html, acedido a 11 Agosto 2011.

[18] CODD E Frank, "A Relational Model of Data for Larged Shared Data Banks", Communications of the ACM, 1970, Vol. 13.

[19] Oracle Corporation, "Getting Started with the JDBC API". [Online] Disponível em: http://download.oracle.com/javase/6/docs/technotes/guides/jdbc/getstart/GettingStartedTOC.fm.html, acedido a 14 Outubro 2011.

[20] OTTINGER J, "TheServerSide.com", 2008. [Online] Disponível em: http://www.theserverside.com/news/ 1363671/What-is-an-App-Server, acedido a 16 Agosto 2011.

[21] Oracle Corporation, "GlassFish Server Open Source Edition 3.1-3.1.1 Release Notes", 2011.

[22] IBM, 2011. [Online] Disponível em: http://www-01.ibm.com/software/webservers/appserv/was/features/, acedido a 16 Agosto 2011.

[23] JENDROCK E, EVANS I, et al., "The Java EE 6 Tutorial", 2011. [Online] Disponível em: http://download.oracle.com/javaee/6/tutorial/doc/, acedido a 19 Agosto 2011.

[24] BISWAS R, ORT E, "The Java Persistence API - A Simpler Programming Model for Entity Persistence", 2006. [Online] Disponível em: http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html, acedido a 26 Agosto 2011.

[25] HEFFELFINGER D R., "Java EE 6 Development with NetBeans 7", Packt Publishing Ltd.; 2011. págs. 135-136.

[26] HALL M, "Managed Beans II – Advanced Features", 2011. [Online] Disponível em: http://courses.coreservlets.com/Course-Materials/pdf/jsf/jsf2/JSF2-Managed-Beans-2.pdf, acedido a 30 Agosto 2011.

[27] BURNS E, SCHALK C, "JavaServer Faces 2.0: The Complete Reference", Capítulo 16, McGraw-Hill; 2010.

[28] HALL M, "Managed Beans I – Classes to Represent Form Info", 2011. [Online] Disponível em: http://courses.coreservlets.com/Course-Materials/pdf/jsf/jsf2/JSF2-Managed-Beans-1.pdf, acedido a 3 Setembro 2011.

[29] BURNS E, SHALK C, "JavaServer Faces 2.0: The Complete Reference", Capítulo 6, McGraw Hill; 2010.

[30] "API Directory Popularity". [Online] Disponível em: http://www.programmableweb.com/apis/directory/1?sort=mashups, acedido a 24 Setembro 2011.

[31] Google, "Google Calendar - View from other applications". [Online] Disponível em: http://www.google.com/support/calendar/bin/answer.py?answer=37648, acedido a 19 Outubro 2011.

[32] Microsoft, "XML Web Services Basics". [Online] Disponível em: http://msdn.microsoft.com/en-us/library/ms996507.aspx, acedido a 19 Outubro 2011.

[33] TIOBE Software, "TIOBE Programming Community Index for September 2011". [Online] Disponível em: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, acedido a 12 Setembro 2011.

[34] "Mono Project". [Online] Disponível em: http://www.mono-project.com/Main_Page, acedido a 12 Setembro 2011.

[35] PEEL T, MACHÁCEK J, "Technical Guidelines For IT Professionals - Java vs..NET", National Computing Centre Publications, 2011.

[36] NIELSEN J, "Enhancing the explanatory power of usability heuristics", ACM, 1994.

[37] GARETS D, DAVIS M, Healthcare Informatics, "Electronic Patient Records - EMRs and EHRs", 2005.

[38] HIMSS EHR Committee, "HIMSS Electronic Health Record Definitional Model v1.0", 2010.

[39] EHR Scope, 2009. [Online] Disponível em: http://www.ehrscope.com/emr-software-advantages-disadvantages-of-a-web-based-system, acedido a 4 Agosto 2011.

[40] STROETMANN K A., "Electronic health record systems in Europe", 2010; London.

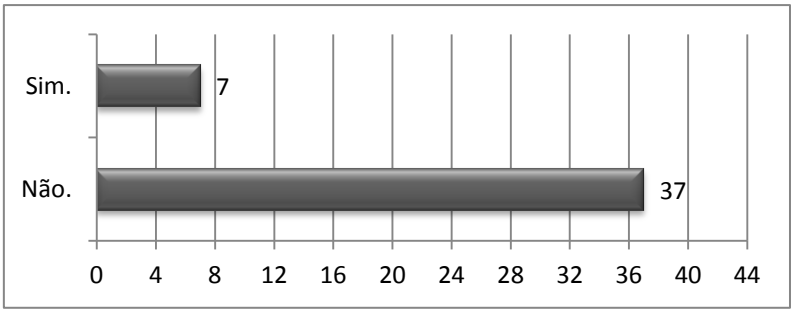
Anexo I

De modo a conhecer melhor as estatísticas de utilização do serviço de consultas gratuitas dos SASUA, e as opiniões sobre alguns aspetos da gestão do mesmo, foi realizado um pequeno inquérito. Este inquérito foi realizado exclusivamente *online*, e foi partilhado através de redes sociais entre alunos e funcionários da Universidade de Aveiro.

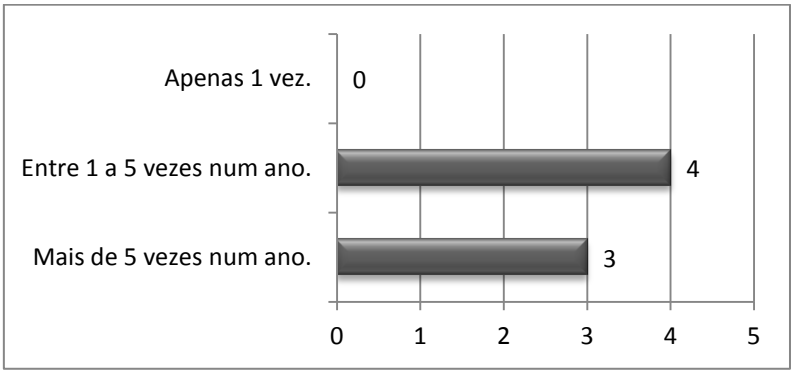
A referir novamente que o objetivo deste inquérito foi apenas o de obter alguns dados estatísticos, não se tratando de forma alguma de um inquérito oficial criado pelos SASUA.

Apresentam-se de seguidas as perguntas presentes no inquérito e os respectivos resultados, num total de 44 respostas.

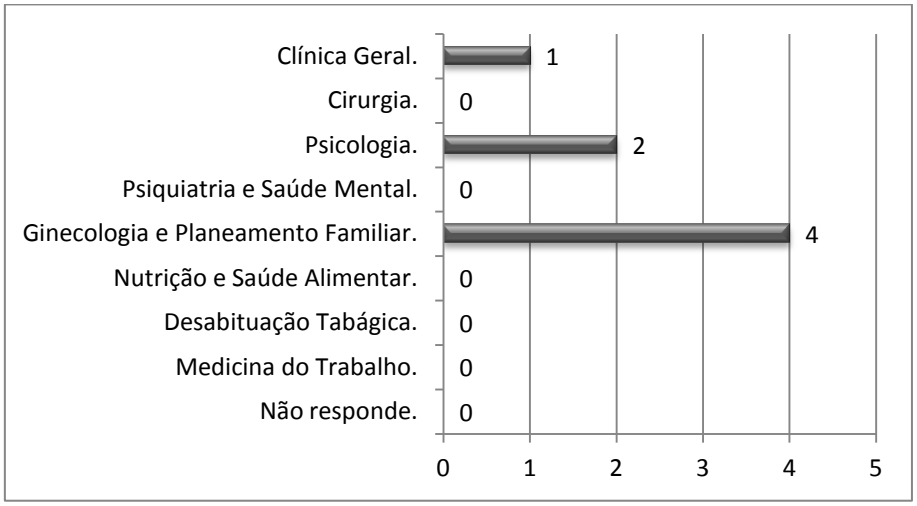
1) Já usufruiu das consultas de especialidade gratuitas oferecidas pelos SASUA?



2) Se sim, com que frequência?



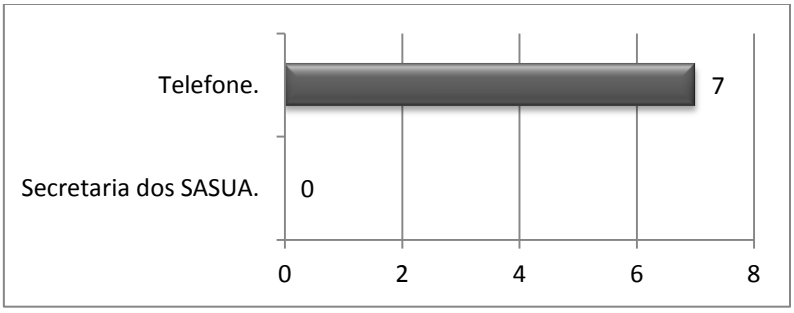
3) Se respondeu afirmativamente à pergunta 1), quais as especialidades médicas oferecidas pelos SASUA que já usufruiu?



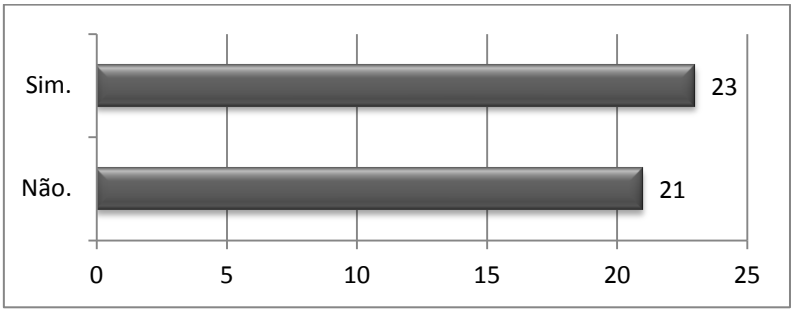
4) Existe alguma especialidade médica que gostasse de ver exercida neste serviço e que ainda não existe? Qual?

- Oftalmologia
- Dermatologia
- Estomatologia
- Medicina Dentária

5) Que método de marcação de consultas dos SASUA já utilizou?



6) Considera as formas disponíveis de marcação destas consultas práticas e funcionais?



7) Qual a sua opinião sobre uma possível plataforma de gestão de consultas online?